TITLE OF INVENTION

METHOD AND APPARATUS FOR ROUTING DATA OVER MULTIPLE WIRELESS NETWORKS

INVENTORS

David L. Whitmore
Christopher James Bogdon
Charles Allan Dick

P17951.S01

METHOD AND APPARATUS FOR ROUTING DATA OVER MULTIPLE WIRELESS NETWORKS

CROSS REFERENCE TO RELATED APPLICATIONS

The present application is related to U.S. Patent Application No. 09/527,014, filed on March 16, 2000, entitled "Apparatus and Method for Intelligent Routing of Data between a Remote Device and a Host System," which is a continuation of U.S. Patent Application No. 08/932,532, filed on September 17, 1997, entitled "Apparatus and Method for Intelligent Routing of Data between a Remote Device and a Host System," which is a continuation-in-part of U.S. Patent No. 5,717,737, issued on April 14, 1997, entitled "Apparatus and Method for Transparent Wireless Communication Between a Remote Device and a Host System," the contents of which are expressly incorporated by reference herein in their entireties.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of wireless communications. More particularly, the present invention relates to routing both IP and non-IP data over multiple wireless networks.

2. Background Information

Currently there are several problems in trying to simultaneously communicate between a client device and a host device over two or more wireless networks. The primary problem is that two network connections require two network addresses, complicating transmitting through the two connections. For example, when an application running on the host device needs to send data to a client device over one of the wireless networks, the host application must determine to which address (i.e.,

through which wireless network) to transmit the data. In known systems, the host application must send data to the first address and then to the second address to determine the appropriate network to use, for example to determine which address is within coverage or which network is down. Moreover, in the known systems, standard host applications (e.g., web browsers) require modification so that they can serially attempt to transmit to the multiple IP addresses.

A second problem with current IP wireless networks is that the network provider assigns its own IP address to each wireless network user. Quite often, this IP address does not coincide with the administrator's LAN IP addressing scheme. Therefore, care is required when managing multiple IP address subnets. Using more than one wireless provider further exacerbates the situation by bringing in additional (most likely inconsistent) addressing schemes. Moreover, several wireless network providers use dynamic IP addressing, raising a new set of security concerns for the network administrator.

Another problem with current scenarios is the inability to mix an IP network such as CDPD with a non-IP network such as a Satellite network, transparently from the application (e.g., a web browser). In the current scenario, the application would have to dynamically change the method for data delivery to the wireless network, requiring modification of the application. Modifying the application typically entails modifying the application's source code, which is often not readily available.

Therefore, a need exists to simplify communications over multiple wireless networks to a single mobile device. Moreover, a need exists for a simple addressing scheme that is independent of wireless network providers' addressing schemes. That is, the network administrator should have the flexibility to implement his own

addressing system. Finally, a need exists for a system that enables IP applications to operate with non-IP networks, without modification of the IP applications.

SUMMARY OF THE INVENTION

In view of the foregoing, the present invention is directed to simplifying the IP addressing scheme of wide area networks including wireless networks. The present invention, which may be embodied as mobile routing software, allows a mobile wireless data user to simultaneously communicate over multiple wireless networks to a local area network.

A mobile routing device is provided that communicates over multiple wireless networks with a Host Network Server residing on a Local Area Network. The mobile routing device also communicates with at least one client device. The mobile routing device includes multiple router network adapters. Each router network adapter interfaces with one of the wireless networks to send and receive data from the wireless network. Each router network adapter has a gateway address, associated with the wireless network, that the Host Network Server uses to send data to the mobile routing device. The mobile router also includes at least one client router network adapter that interfaces with the at least one client device. Each client router network adapter is associated with an end point address that an application uses to send data to the client device. Thus, data is sent to the client device via the Host Network Server, via at least one of the wireless networks, and via the mobile routing device, using only the end point address so that the application sending the data is unaware of the wireless networks used to transport the data and the corresponding gateway addresses.

In one embodiment, each client router network adapter and each router network adapter converts data from an internal protocol format to an outbound format for sending data and converts data from an incoming format in which received data is in the internal protocol format, and monitors the status of the associated wireless network and client communications link. The internal protocol can be Internet Protocol. A Router System Module can also be included for configuring and launching each router network adapter, client router network adapter and a Router Module.

Each router network adapter may send a network control process message to a Router Module indicating whether the associated wireless network is operational. In this case, the Router Module selects one of the wireless networks from candidate wireless networks for data transmission only when the Router Module has received the message indicating that the associated candidate wireless network is operational. In one embodiment, the sending application resides on a Host Application on the Local Area Network.

According to an aspect of the invention, the Router Module generates a Route Registration packet and sends the Route Registration packet to the Host Network Server, when the Router Module has selected a new wireless network. The Route Registration packet includes the gateway address of the new wireless network and the end point addresses that can be reached via the gateway address. Thus, the Host Network Server remains aware of all end point addresses that can be reached via the gateway address contained in the Route Registration packet.

In another embodiment, a second mobile routing device sends data to the mobile routing device via the Host Network Server using the end point address. The

second mobile routing device sends data to the client device via the mobile routing device, via the Host Network Server, and at least one of the wireless networks using only the end point address so that the second mobile routing device is unaware of the wireless networks used to transport the data and the corresponding gateway addresses.

A Router Configuration Module may also be included for reading in configuration data for each router network adapter and for each client router network adapter. The configuration data includes the gateway addresses and the end point addresses. The gateway address may be an IP address when the wireless network is an IP network. The gateway address may be a hardware address when the wireless network is a non-IP network.

A method is provided for routing data to a client device communicating with a mobile routing device via a Host Network Server and at least one wireless network. The method includes identifying the client device and a corresponding end point address; forwarding the data to the Host Network Server using the end point address; and receiving the data at the Host Network Server. The method also includes ascertaining a gateway address corresponding to the end point address. The gateway address is associated with a selected wireless network that was selected for communicating with the mobile routing device corresponding to the client device. The method further includes forwarding the data to the mobile routing device via the selected wireless network using the gateway address; and forwarding the data from the mobile routing device to the client device based upon the end point address.

In one embodiment, the receiving includes receiving the data at a Network Interface from an originating wireless network; saving a source hardware address and an end point hardware address, if the originating wireless network is a non-IP

network; and forwarding the data to a Router Manager. If the originating wireless network is a non-IP network, the method further includes analyzing the source hardware address. If the originating wireless network is an IP network, the method further includes analyzing the source IP address,

According to an aspect of the invention, the analyzing further includes determining whether the source address is present in a route table; updating the route table to reflect that data has been received from the wireless network corresponding to the source address, if the source address is present in the route table; and adding the source address to the route table, if the source address is not present in the route table.

The receiving may also include receiving the data at an IP stack from a Local Area Network; and forwarding the data to a Router Manager. The ascertaining may include determining a subnet that the end point address resides on, and looking up the gateway address in the route table based upon the subnet.

In one embodiment, forwarding the data to the mobile routing device further includes forwarding the data to a Network Interface; translating the data to a format compatible with the wireless network, if the wireless network is a non-IP network; and transmitting the data via the wireless network. If the data cannot be transmitted via the wireless network, the method includes determining if an alternate route to the mobile routing device exists; forwarding the data to an alternate Network Interface, if an alternate route exists; translating the data to a format compatible with the alternate wireless network, if the alternate wireless network is a non-IP network; and transmitting the data via the alternate wireless network. The determining, forwarding, translating and transmitting repeat until the data is successfully transmitted or no alternate routes exist.

According to one aspect of the invention, the forwarding to the client device further includes receiving the data at a router network adapter; translating the data from a format compatible with the wireless network into an IP format, if the wireless network is a non-IP network; determining whether the end point address is known locally; forwarding the data to a client router network adapter, when the address is known locally; and transmitting the data to the client device.

A Host Network Server is provided that communicates with at least one mobile routing device via a plurality of wireless networks. The Host Network Server includes a Router Manager that selects at least one of the wireless networks for data transmission based upon an end point address identifying the mobile routing device. The selected wireless network is identified by a gateway address. The Host Network Server also includes Network Interfaces. Each Network Interface interfaces with one of the wireless networks to send and receive data from the wireless network. Each Network Interface converts data to and from a format associated with the wireless network when the wireless network format is different from an internal format.

The Host Network Server may also include a route table that associates each end point address with at least one gateway address. The Host Network Server determines a wireless network to use for sending data to each end point address based upon a lookup in the route table.

A data routing system is provided for routing data over at least one of a plurality of wireless networks. The system includes a Host Network Server residing on a Local Area Network. The Host Network Server includes a Router Manager that selects at least one of the wireless networks for data transmission based upon an end point address corresponding to a client device associated with the mobile routing

device. The selected wireless network is identified by a gateway address. The Host Network Server also includes host Network Interfaces. Each host Network Interface interfaces with one of the wireless networks to send and receive data from the wireless network.

The system also includes a mobile routing device including router network adapters. Each router network adapter interfaces with one of the wireless networks to send and receive data from the wireless network, and has a gateway address, associated with the wireless network, that the Host Network Server uses to send data to the mobile routing device. The mobile routing device also includes at least one client router network adapter that interfaces with the at least one client device. Each client router network adapter has an end point address that a sending application uses to send data to the client device. Consequently, the sending application sends data to the client device via the Host Network Server, via at least one of the wireless networks and via the mobile routing device, using only the end point address so that the sending application is unaware of the wireless networks used to transport the data and the corresponding gateway addresses.

A computer readable medium storing a computer program is provided for routing data between a Host Network Server and a client device associated with a mobile routing device over at least one of a plurality of wireless networks. The program includes identifying the client device and a corresponding end point address; forwarding the data to the Host Network Server using the end point address; and receiving the data at the Host Network Server. The program also includes ascertaining a gateway address corresponding to the end point address, the gateway address being associated with a selected wireless network that was selected for

communicating with the mobile routing. The program further includes forwarding the data to the mobile routing device via the selected wireless network using the gateway address; and forwarding the data from the mobile routing device to the client device based upon the end point address.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is further described in the detailed description that follows, by reference to the noted plurality of drawings by way of non-limiting examples of preferred embodiments of the present invention, in which like reference numerals represent similar parts throughout several views of the drawings, and in which:

Fig. 1 illustrates a general overview of a remote network controller and mobile data controller in accordance with an aspect of the present invention;

Fig. 2 illustrates a block diagram of the basic components of the remote network controller and mobile data controller of the present invention;

Fig. 3 is a high-level flow chart, according to the present invention, of the outbound data from a wired communication network to a remote device;

Fig. 4 is a high-level flow chart, according to the present invention, illustrating the flow of inbound data from a remote device to a wired communication network;

Fig. 5 illustrates a block diagram of the components of a mobile interface in accordance with the present invention;

Fig. 6 is a flow chart of the processing of an event handler and multithreading dispatcher associated with the mobile interface of the present invention;

Fig. 7 is a flow chart for indicating the process flow of a process initialization module associated with the mobile interface of the present invention;

Fig. 8 is a flow chart for indicating the processing of a mobile session manager associated with the mobile interface of the present invention;

Fig. 9 is a flow chart of the processing steps for an inbound data event handler associated with the mobile interface of the present invention;

Fig. 10 is a flow chart of the processing steps for an outbound data event handler associated with the mobile interface of the present invention;

Fig. 11 is a flow chart of the processing steps for a process termination module associated with the mobile interface of the present invention;

Fig. 12 is a flow chart of the processes associated with a host data controller interface module associated with the mobile interface of the present invention;

Fig. 13 is a block diagram of the various components of a host data controller in accordance with an aspect of the present invention;

Fig. 14 is a block diagram of the components comprising a service interface according to the present invention;

Fig. 15 is a flow chart of the processing of an event handler and multithreading dispatcher associated with the service interface of the invention;

Fig. 16 is a flow chart describing the process flow of a process initialization module associated with the service interface of the invention;

Fig. 17 is a flow chart of the processing steps for an inbound data event handler associated with the service interface of the present invention;

Fig. 18 is a flow chart of the processing steps for an outbound data event handler associated with the service interface of the present invention;

Fig. 19 is a flow chart of the processing steps for a process termination module associated with the service interface of the present invention;

Figs. 20, 21, 22, 23A, 23B and 24 are flow charts of the various processes associated with a wired communication network interface module associated with the service interface in accordance with the present invention;

Fig. 25 is a block diagram of the various components of a mobile data controller of the present invention;

Fig. 26 is a block diagram of the various components of a remote gateway according to another aspect of the present invention;

Figs. 27 and 28 illustrate a block diagram of a remote network controller in accordance with still another aspect of the present invention, in which a subsystem synchronization process module is utilized;

Fig. 29 illustrates a general overview of another embodiment of the present invention which includes a mobile router in accordance with an aspect of the present invention;

Fig. 30 illustrates a schematic block diagram of the mobile router in accordance with an aspect of the present invention;

Fig. 31 is an illustration of a block diagram of the functional components of the router in accordance with an aspect of the present invention;

Fig. 32 is an illustration of a block diagram of the switch within the router according to the present invention;

Fig. 33 is an illustration of a flow chart of the processing steps used by the router to initialize and build tables stored in the Router in accordance with an aspect of the present invention;

Fig. 34 is a flow chart of the processing steps used by the router for checking availability of each network interface in accordance with an aspect of the present invention;

Fig. 35 is a flow chart of the processing steps used by the router to account the availability of the channels and the user's configuration in order to decide which channel to use for transporting data in accordance with an aspect of the present invention;

Fig. 36 is a flow chart of the processing steps used by the router for an error handler in accordance with an aspect of the present invention;

Fig. 37 is an illustration of the software architecture of the Router in accordance with an embodiment of the present invention;

Fig. 38 shows an overall system diagram including a Host Network Server, multiple wireless networks, and multiple mobile devices, according to an aspect of the present invention;

Fig. 39 shows a software architecture for a Host Network Server, in accordance with an aspect of the present invention;

Fig. 40 is a flow chart showing an exemplary process executed by the Host Network Server for processing incoming data received on a wireless network, according to an aspect of the present invention;

Fig. 41 shows an exemplary route table, according to an aspect of the present invention;

Figs. 42, 43, and 44 are flow charts showing exemplary logic executed by the Host Network Server for processing outgoing data, according to an aspect of the present invention;

Fig. 45 shows a software architecture for a remote routing device in an initial state, according to an aspect of the present invention;

Fig. 46 shows an exemplary configuration data block, according to an aspect of the present invention;

Fig. 47 shows a software architecture for the remote routing device at a later state, according to an aspect of the present invention; and

Fig 48 shows an exemplary route registration packet, according to an aspect of the present invention.

DETAILED DESCRIPTION

The mobile routing software of the present invention provides a mobile device with a single IP address, even though the mobile device is connected to multiple wireless networks. The flexibility of the routing software facilitates routing between multiple mobile devices as well as over multiple networks. There are no restrictions as to the types of wireless networks. Therefore, several types of networks, both IP and non-IP based, can be mixed. IP addressing is handled automatically because the end point IP address of the mobile computer is identified by a single IP address, rather than the multiple IP addresses associated with the wireless networks. Moreover, this end point IP addressing scheme is determined by the network administrator rather than by the wireless network provider.

Referring now to the accompanying drawings, Fig. 1 illustrates a general overview of a remote network controller and a mobile data controller in accordance with an aspect of the present invention. In Fig. 1, a wired communication network 10 is shown as a host network system having communications controllers 15 and locally-attached devices 12. The wired communication network 10 may be, for

example, a Token Ring network or an Ethernet Local Area Network (LAN). The locally-attached devices 12 may include a personal computer, a workstation, a printer or network server, and reside at a plurality of dispersed locations. According to the present invention, a remote network controller 20 may also be provided which logically resides on the wired communication network 10 and acts as a protocol-appropriate communications controller to send and receive data to and from the communications network 10 and one or more remote or mobile devices 52. For purposes of illustration, only one of the remote devices 52 is shown in Fig. 1.

Remote devices 52 communicate via a mobile data controller 54 and a wireless radio-frequency (RF) communications link 55 created by the user's radio infrastructure 56 to the remote network controller 20. The mobile data controller 54 may convert asynchronous data from the remote device 52 into an appropriate protocol format of the radio infrastructure 56. In accordance with an aspect of the present invention, the remote devices 52, although not physically connected to the wired communication network 10, are logically connected to the wired communication network 10 through the radio infrastructure 56 and the remote network controller 20 and are indistinguishable from locally-attached devices 12. The remote devices 52 may be, for example, a laptop computer, personal digital assistant (PDA), a credit card reader, or a global positioning system (GPS) receiver. The radio infrastructure 56 may comprise a conventional point to point or trunking radio system.

The logical connection created by the remote network controller 20 between the remote device 52 and the wired communication network 10 is "transparent" to the user of the remote device 52, and to the wired communication network 10. In accordance with an aspect of the invention, the remote network controller 20 takes

data transported by the radio infrastructure 56, irrespective of the format protocol of the radio infrastructure, and converts the data into a format protocol recognized by the wired network 10. Similarly, the remote network controller 20 of the present invention takes data from the wired network 10 and converts the data into a format protocol recognized by the radio infrastructure 56. Accordingly, the user of the remote device 52 does not have to perform any additional steps to send and receive data to and from the wired communication network 10, and the wired communication network 10 does not have to perform any additional steps to send and receive data to and from the remote device 52. The user of the remote device 52 interacts with the wired communication network 10 in a similar manner as a user of the locally-attached devices 12. Similarly, the wired communication network 10 interacts with the remote device 52 in a similar manner as the wired communication network interacts with the locally-attached devices 12.

Referring now to Fig. 2, there is illustrated a block diagram of the basic components of the remote network controller 20 of the present invention. Each component of the remote network controller 20 will be generally described for introductory purposes and will later be described in greater detail below with reference to the accompanying drawings. The various components of the host data controller 22 and the mobile data controller 54 will also be discussed hereinafter with reference to Figs. 13 and 25, respectively.

As shown in Fig. 2, the remote network controller 20 may comprise a service interface 30, a mobile interface 24, an interprocess communication manager 28, a control process module 26, and a console interface 34. The remote network controller 20 may be implemented through a collection of software program modules and

hardware components working cooperatively. The remote network controller 20 itself may run on a standard platform, such as a personal computer (PC) equipped with a commercially available processor or multi-processor, e.g., an Intel or Motorola based processor or multi-processor, and a commercially available operating system, such as an MS-DOS or UNIX based operating system. The remote network controller 20 may also contain an Ethernet controller or suitable network controller card depending on the wired communication network 10. In addition, the remote network controller 20 may include random access memory and physical storage media including hard disk and tape storage devices.

The wired communications network 10 is connected to the remote network controller 20 by the service interface 30. The service interface 30 handles all network connections. If several wired communications networks 10 are present, one or more service interfaces 30 may be provided to handle wired network connectivity. The service interface 30 connects to an interprocess communication manager 26. The interprocess communication manager 28 manages all inter-process message routing within the remote network controller 20. One or more mobile interfaces 24 may also be provided to handle connectivity with the radio infrastructure(s) 56. Each mobile interface 24 is also connected to the interprocess communication manager 28. The control process module 26 of the remote network controller 20 is provided to process management functions and data integrity. The control process module 26 is connected to the interprocess communication manager 28 and the console interface 34. The console interface 34 allows for user configuration and reporting of data.

As further illustrated in Fig. 2, the remote network controller 20 may be connected to a host data controller 22. One or more host data controllers 22 may be

provided for connecting the remote network controller 20 to specific radio infrastructures 56, e.g., a Motorola trunked radio. The host data controller 22 may be connected to the mobile interface 24 of the remote network controller 20.

In the field, the remote device 52 is connected to the mobile data controller 54 which, in turn, is connected to the radio infrastructure 56 for transmitting and receiving data. The mobile data controller 54 is responsible for connecting the remote device 52 to the radio infrastructure 56 and to provide protocol-independent asynchronous serial data transfer to and from the remote device 52.

In order to provide transparent data transportation, whereby the network protocols and the protocols of the radio infrastructure 56 are transparent or invisible to the user, inbound asynchronous data from the remote device 52 is collected and transported to the wired communication network 10 in packets over the radio infrastructure 56. The data is sent using the existing protocols of the radio infrastructure 56. The remote network controller 20 accepts the data and encapsulates it into the appropriate protocol used by the wired communication network 10. The data is passed to the wired communication network 10 in a similar fashion for passing data from any of the other locally-attached devices 12. Similarly, outbound data to the remote device 52 from the wired communication network 10 is removed from the network protocol by the remote network controller 20. The remote network controller 20 then encapsulates the data into the appropriate protocol associated with the radio infrastructure 56 and sends the data over the radio infrastructure 56 to the mobile data controller 54. Upon receipt of the data, the mobile data controller 54 removes the data from the radio infrastructure protocol and asynchronously sends the data to the remote device 52.

In accordance with the present invention, multiple wired networks 10 with different protocols may be linked to multiple RF environments in any combination by incorporating the remote network controller and mobile data controller of the present invention.

Fig. 3 is a high-level flow chart for transporting outbound data from the wired communication network 10 to the remote device 52. As shown in Fig. 3, when data is to be sent to the remote device 52, the service interface 30 of the remote network controller 20 accepts data from the wired communication network 10 at step 500. The service interface 30 then converts the data from the protocol used by the wired communication network 10 and encapsulates it into an internal protocol used by the remote network controller 20 at step 502. In addition, the service interface 30 may receive routing information from the wired communication network 10 as to what remote device 52 the data is to be passed, e.g., a network address or device identifier of the remote device 52.

At step 504, the service interface 30 forwards the data to the interprocess communication manager 28. The interprocess communication manager 28 accepts the data at step 506 and, at step 508, places the data in a queue for the appropriate destination mobile interface 24. The destination mobile interface 24 may depend on the radio infrastructure 56 employed by the user. The outbound data that is to be passed from the interprocess communications manager 28 to the mobile interface 24 may be encapsulated in an internal protocol of the remote network controller 20, along with routing information to specify the remote device 52 to which the data is to be sent. At step 510, the interprocess communication manager 28 notifies the mobile interface 24 that the data to be sent to the remote device 52 is queued for the mobile

interface. The particular mobile interface 24 that the data is queued for depends on the particular radio infrastructure 56 employed to communicate with the destination remote device 52. At step 512, the mobile interface 24 requests that the queued data be sent from the interprocess communication manager 28. The mobile interface 24 may request data when it is free to send the data to a destination remote device 52 and not handling another process. At step 514, the mobile interface 24 accepts the queued data from the interprocess communication manager 28. Thereafter, at step 516, the mobile interface 24 determines, based on the queued data, the destination node address of the remote device 52 to which the data is to be sent. At step 518, the mobile interface 24 forwards the data to the appropriate host data controller 22 so that it may be sent over the radio infrastructure 56 at step 520. According to an aspect of the present invention, the host data controller 22 may receive the data, remove it from the internal protocol and encapsulate the data into a packet determined by the protocol used by the radio infrastructure 56. The packet of data may be broadcasted over the radio infrastructure 56 so as to enable the host data controller 22 to communicate with multiple mobile data controllers 54 simultaneously. The broadcasted data packet may include the identification of the specific mobile data controller 54 to which the packet is to be delivered, so that only uniquely identified mobile controller(s) may accept the packet.

Referring again to Fig. 3, at step 522, the mobile data controller 54 receives the data from the remote radio infrastructure 56 and decodes the data. The data packet, once received by the mobile data controller 54, is accepted and the data is removed from the packet. At step 524, the mobile data controller 54 validates the data and, at step 526, sends an acknowledgment or rejection message to the host data controller

22 via the radio infrastructure 56. According to the present invention, the remote network controller 20 and the host data controller 22 may be responsible for ensuring the integrity of the data transported over the radio infrastructure. As such, an error detection/retry mechanism may be employed to detect and correct data transmission errors. After the integrity of the data is verified, the mobile data controller 54 at step 528 will forward the data to the remote device 52. The data may be asynchronously transferred to the remote device 52 through a serial connection.

Fig. 4 is a high-level flow chart illustrating the processing of inbound data from the remote device 52 to the wired communication network 10. At step 550, the mobile data controller 54 accepts data from the remote device 52. At step 552, the mobile data controller 54 formats and sends the data to the remote network controller 20 via the radio infrastructure 56, which may comprise a modem. The data may be transmitted using the appropriate protocol of the radio infrastructure 56. The data may be modulated within the mobile data controller 54 prior to transmission via the radio infrastructure 56. The mobile data controller 52 may place the data from the remote device 52 into packets to be sent over the radio infrastructure 56. The packet size can be determined by one of three methods. The first is a maximum packet size. Once an upper limit of data is accumulated, the mobile data controller 54 may send the packet of information to the host data controller 22. For example, once 256 bytes of data are collected, the data may be sent by the radio infrastructure 56 over the RF communications link 55. The second method is a maximum time to wait before sending data. In this case the mobile data controller 54 will send a packet after waiting a predetermined period of time, no matter how much data is accumulated.

The third method involves the mobile data controller 54 detecting a predefined "end-of-packet" character which causes all accumulated data to be transmitted.

At step 554, the host data controller 22 receives and decodes the data packet from the protocol of the radio infrastructure 56. Generally, the data arrives as a packet of a predetermined size. At step 556, the host data controller 22 validates the data and, thereafter, sends at step 558 an acknowledgment or rejection message to the mobile data controller 54 based on the validation process. According to an aspect of the present invention, the host data controller 22 may determine if the transmitted data packet is correct, or in error. The host data controller 22 may also determine if the data packet has arrived in the proper sequence, and that the packet is not a duplicate. As discussed above, the inbound data may be removed from the packet and encapsulated in the internal protocol used by the remote network controller 20. The internal protocol may contain additional information, such as the identification of the mobile data controller 54 which sent the information.

At step 560, the host data controller 22 forwards the data to the mobile interface 24. The mobile interface 24 accepts the data from the host data controller 22 at step 562. The mobile interface 24 validates the address of the source of the data (e.g., the particular mobile data controller 54 or remote device 52) at step 562. At step 566, the mobile interface 24 forwards the data to the interprocess communication manager 28, which accepts the data at step 568. The mobile interface may also pass the routing information specifying the remote device 52 from which the data originated. At step 570, the interprocess communication manager 28 places the data into a queue for the destination service interface 30. The particular destination service interface 30 will depend upon which wired communication network 10 the data is to

be delivered. Included in the information which is passed to the service interface 30 is the destination address (i.e., the communication network 10 to which the data is to be delivered). At step 572, the service interface 30, when available to handle data, requests the data from the interprocess communication manager 28. The service interface 30 accepts the data at step 576 and converts the data into an appropriate form, i.e., protocol, usable by the wired communication network 10 at step 578. As a result, the data may be passed to the hardware device (e.g., an Ethernet controller) using the protocol required by the wired communication network 10. This configuration allows any existing network interface card to be used in conjunction with the remote network controller 20, because the data is placed into the appropriate network protocol by the service interface 30 before it is transmitted to the wired network. At step 580, the service interface 30 forwards the data to the wired communication network 10.

The validation process of the outbound data depicted in Fig. 3 and inbound data depicted in Fig. 4 does not depend on the type of wired communication network 10 employed by the user. Through a single validation process performed by the host data controller 22 and the mobile data controller 54 (see steps 524 and 526 in Fig. 3 and steps 556 and 558 in Fig. 4), the integrity of the data transmitted from the wired communication network 10 to the remote device 52 through the radio infrastructure 56 is ensured. This validation process may include, for example, an error detection and retry mechanism to detect errors and to cause (when necessary) the retransmission of the data.

Referring now to Fig. 5, there is illustrated a block diagram of the basic components of the mobile interface 24 of the remote network controller 20 of the

present invention. As noted above, the mobile interface 24 is responsible for interfacing the remote network controller 20 with the host data controller 22 and the radio infrastructure 56. The mobile interface 24 may be a software interface that records statistical information related to inbound and outbound data. The mobile interface 24 may also be responsible for error detection and correction, and establishing and managing the mobile data sessions with the remote devices 52. The number of mobile interfaces 24 provided in the remote network controller 20 depends on the number of different types of radio infrastructures 56 employed by the user. Each type of radio infrastructure 56 may have its own associated mobile interface 24.

As shown in Fig. 5, the mobile interface 24 may include an event handler and multithreading dispatcher 60, a process initialization module 62, a mobile session manager 64, an inbound data event handler 66, an outbound data event handler 68, a process termination module 70 and a host data controller interface module 72. The event handler and multithreading dispatcher 60 may contain high-level logic and be used to control the overall execution flow of the mobile interface 24. The process initialization module 62 may be utilized to acquire resources and establish the operation environment for the mobile interface 24 process. The process initialization module 62 may also be provided to initialize the host data controller 22.

According to the present invention, the mobile session manager 64 may be provided to control the communications environment between the mobile data controller 54 and the host data controller 22. The inbound data event handler 66 responds to signals from the host data controller 22 indicating that inbound data is available and preprocess session control information. The outbound data event handler 68 is provided to respond to signals from the interprocess communication

manager 28 indicating that outbound data is available or that a session control function is required. The process termination module 70 functions to release previously-acquired resources and terminate the mobile interface 24 process efficiently. The host data controller interface module 72 handles low-level interaction with the associated host data controller(s) 22.

The process flow of the event handler and multithreading dispatcher 60 will now be described with reference to Fig. 6. At step 600, the process begins when the remote network controller 20 is powered up and initialized. At step 602, the process initialization module 62 is invoked (described below with reference to Fig. 7). At step 604, the event handler and multithreading dispatcher 60 waits for an event (e.g., receipt of inbound data) to occur. While the event handler and multithreading dispatcher 60 waits for an event to occur, mobile interface 24 may be placed in a "sleep" mode to conserve processor resources. At step 606, once an event occurs, the event handler and multithreading dispatcher 60 determines if it is a recognized event. If the event handler and multithreading dispatcher 60 determines it is not a recognized event at step 606, processing returns to step 604. If, however, the event handler and multithreading dispatcher 60 determines that the event is a recognized event at step 606, then processing continues at step 608, where the event handler and multithreading dispatcher 60 determines if the data was received from the host data controller 22.

At step 608, if the event handler and multithreading dispatcher 60 determines the data was received from the host data controller 22, the event handler and multithreading dispatcher 60 invokes the inbound data event handler 66, at step 614 (described below with reference to Fig. 9) and processing continues at step 604. If

at step 608 the event handler and multithreading dispatcher 60 determines that the data was not received from the host data controller 22, then the event handler and multithreading dispatcher 60 determines whether the data was received from the service interface 30 at step 610.

If the event handler and multithreading dispatcher 60 at step 610 determines that the data was received from the service interface 30, then at step 616 the outbound data event handler 68 is invoked (described below with reference to Fig. 10) and processing returns to step 604. If the event handler and multithreading dispatcher 60 at step 610 determines that the data was not received from the service interface 30, then at step 612 the event handler and multithreading dispatcher 60 determines if there is a process termination request.

If, at step 612, the event handler and multithreading dispatcher 60 determines there is a process termination request, then at step 618, the process termination module 70 is invoked (described below with reference to Fig. 11). If, at step 612, the event handler and multithreading dispatcher 60 determines that there is not process termination request, then processing continues at step 604 to wait for another event.

Referring now to Fig. 7, there is illustrated an exemplary flow chart for indicating the process flow of the process initialization module 62 of Fig. 5. At step 620, the interprocess communications interface is setup. At step 622, the operating environment parameters are parsed and processed. This includes the host data controller 22 parameters referenced in steps 626, 632 and 634 below. At step 624, memory is allocated for the session and other tables contained within the mobile interface 24, which are used to control data flow and other operations. At step 626, the host data controller 22 parameters are accessed. At step 628, a path to the host

data controller 22 port is opened. At step 630, the host data controller 22 then is prevented from monitoring for an event from the remote device(s) 52. Step 630 prevents erroneous transmissions that may arise if the host data controller 22 attempts to monitor a remote device 52 before the initialization process is complete. At step 632, the host data controller 22 communication parameters are set. At step 634, the communication parameters are downloaded to the host data controller 22. After the initialization processes of steps 632 and 634 are completed, the host data controller 22 at step 636 is enabled to monitor the remote device(s) 52. At step 638, the entire initialization procedure is complete and processing returns to step 604 in Fig. 6.

Referring now to Fig. 8, there is illustrated an exemplary flow chart describing the logic flow of the mobile session manager 64 of Fig. 5. At step 640, the mobile session manager 64 handler is entered from the event handler and multithreading dispatcher 24 when remote data is detected. At step 642, the remote identifier of the remote device 52 is looked up in a session table. At step 644, the mobile session manager 64 determines if the remote identifier was found in the session table. If the mobile session manager 64 determines that the remote identifier was found, the address is returned from the session table at step 646.

If the mobile session manager 64 does not find the remote identifier at step 644, then at step 648 the mobile session manager 64 attempts to authenticate the remote identifier. At step 650, the mobile session manager determines if the authentication is successful. If at step 650 the authentication is successful, then at step 656 the host data controller 22 is instructed to connect to the remote device 52 based on the remote identifier. After the host data controller 22 is connected to the remote device 52, the appropriate service interface 30 is invoked at step 658. At step

660, processing is complete. If at step 650, the authentication was not successful, the remote data is ignored at step 652, and a null session table entry address is returned by the mobile session manager 64.

Referring now to Fig. 9, there is illustrated an exemplary flow chart of the processing steps of the inbound data event handler 66 of Fig. 5. At step 662, the inbound data event handler is invoked (from step 614 in Fig. 6). At step 664, the remote identifier of the remote device 52 is checked against the session table. At step 666, it is determined whether the inbound data event handler 66 found the remote identifier in the session table. If at step 666, the remote identifier is not found in the session table, the data is ignored and processing continues at step 604 in Fig. 6. If the remote identifier is found in the session table at step 666, the data is sent to the service interface 30 at step 670. Processing then continues at step 604 in Fig. 6.

Referring now to Fig. 10, there is illustrated an exemplary flow chart of the processing steps of the outbound data event handler 68 of Fig. 5. At step 672, the outbound data event handler is invoked (from step 616, Fig. 6). At step 674, the session table is checked for the outbound data remote identifier. At step 676, it is determined if the outbound data event handler 68 found the remote identifier in the session table. If at step 676, the remote identifier is not found in the session table, an error is logged and the data message is ignored. Processing then continues at step 604 in Fig. 6. If the remote identifier is found in the session table at step 676, the data is sent to the remote device 52 as a single packet at step 680. Processing then continues at step 604 in Fig. 6.

Referring now to Fig. 11, there is illustrated an exemplary flow chart of the processing steps of the process termination module 70 of Fig. 5. At step 682, the

process termination module 70 is invoked (from step 618 in Fig. 6). At step 684, the process termination module 70 determines if there are any active remote sessions. If, at step 684, it is determined by the process termination module 70 that there are no active sessions, then at step 686 all files are closed and the mobile interface 24 terminates. If, however, it is determined by the process termination module 70 that there are active sessions, then at step 688 all of the active sessions are issued a disconnect request. At step 690, the process termination module waits for all active sessions to terminate. Once all active sessions have terminated at step 690, then all files are closed and the mobile interface 24 terminates at step 686.

Referring now to Fig. 12, there is illustrated an exemplary flow chart of the processes associated with the host data controller interface module 72 of Fig. 5. The host data controller interface module 72 consists of a number of discrete functions (e.g., Initialize, Command, Send Data, and Receive Data) which are called when needed by the mobile interface 24 and share common information about the host data controller 22. The host data controller interface module 72 may access the host data controller 22 via a serial communications port which is assigned to the mobile interface 24 and remains fixed when the remote network controller 20 is in operation.

A host data controller 22 initialize routine begins at step 692. The initialization routine may be initiated in accordance with step 602 (see Fig. 6) and steps 632 and 634 (see Fig. 7). At step 694, the serial communications port is accessed and setup. Thereafter, at step 696, the port handle and status is saved to be used by other processes within the host data controller interface module 72.

A host data controller 22 command routine begins at step 698. The command routine may be initiated upon the occurrence of a recognized event (see, e.g., step 604

in Fig. 6) so that the appropriate control or operation commands may be sent to the host data controller 22. At step 700, the host data controller 22 is placed into a command mode. At step 702, a command (e.g., disconnect or receive) is issued to the host data controller 22 based on the event that is recognized. At step 704, the host data controller interface module 72 awaits a confirmation of acceptance of the command from the host data controller 22. At step 706, the result of the command is returned to the host data controller interface module 72.

A host data controller 22 send data routine begins at step 708 and may be initiated from step 680 in Fig. 10. The send data routine is initialized so that data may be sent to the appropriate remote device 52. First, the physical identification of the remote device 52 is determined at step 710. Thereafter, the data to be sent to the remote device 52 is placed into a packet at step 712, and sent to the host data controller 22 at step 714.

At step 716, a host data controller 22 receive data routine is initiated in accordance with step 670 in Fig. 9. The receive data routine is initiated so that data from the remote device 52 may be received by the remote network controller 20. At step 718, data is accumulated within the host data controller 22 receive data routine (see, Fig. 12, step 716) until a full packet of information is received. Thereafter, at step 720, the packet is identified as either session oriented or monitor oriented data. The identified data packet is then returned, at step 722, to the mobile interface 24 and sent to the wired communication network 10 via the remote network controller 20.

Referring now to Fig. 13, in accordance with an aspect of the present invention, there is illustrated a block diagram of the basic components of the host data controller 22 (see, e.g., Fig. 2) of the present invention. The host controller 22 may

be physically connected external to the remote network controller 20 via the mobile interface 24. The host data controller 22 is specifically designed to convert the radio infrastructure 56 protocol to the internal protocol of the remote network controller 20. Typically, one host data controller 22 may be connected to each mobile interface 24; however, one or more host data controllers 22 may be connected for redundancy and greater reliability.

As shown in Fig. 13, the host data controller 22 may comprise an RF communications interface module 80, a remote network controller communications interface module 78, and a configuration and monitoring module 82. The host data controller 22 may comprise any combination of hardware and software to perform the functions described herein. For example, the host data controller 22 may comprise a commercially available processor or multi-processor with overlying application software. The software running in the host data controller 22 may be written in Z80 or other appropriate high-level language (e.g., Pascal). The host data controller 22 may also contain a plurality of serial ports for communicating with other devices.

The remote network controller communications interface module 78 is connected to the mobile interface 24 of the remote network controller 20 and is responsible for sending and receiving data to and from the remote network controller 20. A subsystem port 76 (e.g., an RS-232 adapter) may be used to connect the remote network controller communications interface module 78 to the mobile interface 24 of the remote network controller 20. If the host data controller 20 is connected to more than one remote network controller, then additional subsystem port connection(s) may also be provided to connect to the interface module 78 to the additional remote network controllers. The remote network controller communications interface

module 78 sends health and status information regarding the host data controller 22 to the mobile interface 24. This information informs the remote network controller 20 that the host data controller 22 is operational and accepting data.

The configuration and monitoring module 82 is specific to the type of radio infrastructure 56 employed. Software parameters, such as the number of subsystem ports, how often to send health and status requests, and a list of mobile data controllers 54 to which the host data controller 22 can communicate, may be set and stored in the configuration and monitoring module 82. The configuration and monitoring module 82 can also accumulate statistics which are passed to the mobile interface 24.

In order to diagnose potential system errors in the host data controller 22, the remote network controller 20 may test and analyze the host data controller 22 over a diagnostic port (not shown) to determine a cause of the system failure or error. The diagnostic port may be used not only to determine if the host data controller 22 is operational, but also to configure software parameters particular to the type of radio infrastructure 56. These parameters can be changed to communicate with a different radio infrastructure 56 type as necessary.

The RF communications interface module 80 is responsible for sending and receiving the radio-frequency transmissions. The RF communications interface module 80 is specific to the radio infrastructure 56 in use and is connected to the radio infrastructure 56 by a communication line 57. Again, because the host data controller 22 is designed to integrate with an existing radio infrastructure 56, each host data controller 22 is software configured to work with many different types of radio infrastructure 56 protocols for flexibility. The host data controller 22 may be

designed to be plugged into the remote network controller 20, connecting to the mobile interface 24, and can simply be exchanged with a different host data controller 22, or reprogrammed depending on the radio infrastructure 56 employed. Host data controllers 22 may be configured so as to be compatible with, for example, conventional point-to-point radio systems, conventional repeater-based radio systems, LTR Trunking, Motorola Trunking, Ericsson (EDACS) Trunking - Voice Path, EDACS RDI Trunking - Data Path, and EDACS IMC Voice Path radio infrastructures.

Referring to Fig. 14, there is illustrated a block diagram of the components comprising the service interface 30 (see Fig. 2) of the present invention. The service interface 30 is responsible for communicating to and from the wired communication network 10. The service interface 30 is concerned only with the software level protocols of the wired communication network 10. The hardware interface to the wired communication network 10 is accomplished by a known network control card, such as an Ethernet controller or a Token-Ring controller.

The number of service interface 30 connections to the wired communication network 10 is dictated by the type of wired communication network 10. If the wired communication network 10 uses asynchronous data transfer, there will be one service interface 30 for every entry point, e.g., serial port, into the wired communication network 10. In a local area network (LAN) environment, each service interface 30 may handle a variety of different network addresses. A different service interface 30 may be used for each type of wired communication network 10.

As shown in Fig. 14, the service interface 30 may include an event handler and multithreading dispatcher 90, a process initialization module 92, an inbound data

event handler 94, an outbound data event handler 96, a process termination module 98 and a wired network interface module 100. The event handler and multithreading dispatcher 90 may contain high-level logic and be used to control the overall execution flow of the service interface 30. The process initialization module 92 acquires resources and establishes the operation environment of the service interface 30 process. The inbound data event handler 94 responds to signals from the interprocess communication manager 28 that inbound data is available and preprocess session control information. The inbound data event handler 94 may also handle asynchronous timer events. The outbound data event handler 96 is provided to respond to signals from wired communication network interface module 100 that outbound data is available or that a timer event has occurred. The process termination module 98 functions to release previously-acquired resources and terminate the service interface 30 process gracefully. The wired communication network interface module 100 handles low-level interaction with the associated wired communication network transport mechanism, i.e., communication protocol, being used.

An exemplary process flow of the event handler and multithreading dispatcher 90 (see Fig. 14) of the present invention will now be described with reference to Fig. 15. At step 800, the process begins when the service interface 30 is powered up and initialized. At step 802, the process initialization module 92 is invoked (described below with reference to Fig. 16). At step 804, the event handler and multithreading dispatcher 90 waits for an event to occur in response, e.g., to network or remote device activity. While the event handler and multithreading dispatcher 90 waits for an event to occur, the service interface 30 may be placed in a "sleep" mode to conserve processing power. At step 806, once an event occurs, the event handler and

multithreading dispatcher 90 determines if it is a recognized event. Recognized events may include Initialize, Send Data, Receive Data and/or Terminate. If the event handler and multithreading dispatcher 60 determines it is not a recognized event at step 806, then processing returns to step 804. If the event handler and multithreading dispatcher 90 recognizes the event at step 806, then processing continues at step 808, where the event handler and multithreading dispatcher 90 determines if the data was received from the host communication network 10.

At step 808, if the event handler and multithreading dispatcher 90 determines the data was received from the host wired communication network 10, the event handler and multithreading dispatcher 90 invokes the inbound data event handler 94, at step 814 (described below with reference to Fig. 17) and, thereafter, processing continues at step 804. If at step 808 the event handler and multithreading dispatcher 90 determines that the data was not received from the host wired communication network 10, the event handler and multithreading dispatcher 90 then determines if the data was received from the mobile interface 24 at step 810.

If the event handler and multithreading dispatcher 90 determines at step 810 that the data was received from the mobile interface 24, then at step 816 the outbound data event handler 96 is invoked (described below with reference to Fig. 18) and, thereafter, processing continues at step 804. If the event handler and multithreading dispatcher 90 at step 810 determines that the data was not received from the mobile interface 24, then at step 812 the event handler and multithreading dispatcher 90 determines if there is a process termination request.

If, at step 812, the event handler and multithreading dispatcher 90 determines there is a process termination request, then at step 818 the process termination module

98 is invoked (described below with reference to Fig. 19). However, if at step 812 the event handler and multithreading dispatcher 90 determines that there is not process termination request, then processing returns to step 804 to wait for another event.

Referring now to Fig. 16, there is illustrated an exemplary flow chart describing the process flow of the process initialization module 92 (see, e.g., Fig. 14) of the present invention. At step 820, the interprocess communications interface is setup when the service interface 30 is started or powered up. At step 822, the operating environment parameters are parsed and processed (i.e., the parameters of the operating environment are processed individually). At step 824, any resources required (e.g., memory) are acquired. Thereafter, at step 826, the wired communication network interface module 100 is invoked (see Figs. 20-24 discussed below). As discussed below, the wired communication network interface module 100 may include several procedures associated with initializing the connectivity with the wired communication network 10, reading data from the wired communication network 10, writing data to the wired communication network 10, and terminating connectivity with the wired communication network 10. In accordance with an aspect of the present invention, a unique set of procedures may be provided by the wired communication network interface module 100 for each type of wired communication network 10. For example, a unique set of procedures may be provided for networks utilizing transparent asynchronous communications, TCP/IP stream sockets, Vehicle Location Reporting Facilities, Bidirectional Messaging Facilities, or Credit Card Verification Facilities.

After the wired communication network interface module 100 is invoked, the results of the previous operations performed at step 826 are sent at step 828 to the

mobile interface 24. At step 830, it is determined by the process initialization module 92 if the wired communication network interface module 100 was successfully invoked at step 826. If it is determined at step 830 that the wired communication network interface module 100 was successfully invoked, then the initialization is complete at step 832, and processing returns to step 804 in Fig. 15. If, however, it is determined at step 830 that the wired communication network interface module was not successfully invoked, then the service interface 30 is terminated at step 834.

Referring now to Fig. 17, there is illustrated an exemplary flow chart of the processing steps for the inbound data event handler 94 (see, e.g., Fig. 14) of the present invention. At step 836, the inbound data event handler is invoked (from step 814 in Fig. 15). At step 838, the data portion of the message sent by the wired communication network 10 is extracted. At step 840, the service interface 30 requests a packet of data from the mobile interface 24. At step 842, after the packet is accepted by the service interface 30, data is sent to the appropriate destination via the wired communication network 10. The inbound data event handler 94 then waits for a disconnect request at step 844. If the inbound data event handler 94 receives a disconnect request at step 844, then a disconnect command is sent to the mobile interface 24 at step 846. Processing then continues at step 804 in Fig. 15. If, however, the inbound data event handler 94 does not receive a disconnect request at step 844, then the request received is ignored at step 848 and, thereafter, processing then continues at step 804 in Fig. 15.

Referring now to Fig. 18, there is illustrated an exemplary flow chart of the processing steps for the outbound data event handler 96 (see Fig. 14) of the present invention. At step 850, the outbound data event handler is invoked (from step 816 in

Fig. 15). At step 852, the outbound data event handler 96 determines if there is a request (e.g., a disconnect request from the wired communication network 10 or the remote device 52). If there is a request, then processing continues at step 856. However, if there is presently not a request, then at step 854 the outbound data from the network 10 is sent to remote device 52 via the mobile interface 24. At step 865, the outbound data event handler 96 then determines if a disconnect request has been received. If the outbound data event handler 96 receives a disconnect request at step 856, then a disconnect command is sent to the mobile interface 24 at step 860 by the interprocess communication manager 28 (see, e.g., Fig. 2). Processing then continues at step 804 in Fig. 15. If, however, the outbound data event handler 96 does not receive a disconnect request at step 856, then the request received is ignored at step 862 and processing then continues at step 804 in Fig. 15.

Referring now to Fig. 19, there is illustrated an exemplary flow chart of the processing steps for the process termination module 98 (see Fig. 14) of the present invention. At step 864, the process termination module 98 is invoked (from step 818 in Fig. 15). At step 866, the wired communication network interface module 100 connection is closed. Thereafter, at step 868, the processes associated with the service interface 30 are terminated.

Referring now to Figs. 20-24, there are illustrated exemplary flow charts of the various processes that may be performed by the wired communication network interface module 100 of the present invention. The wired communication network interface module 100 may consist of a number of discrete functions which provide the service interface 30 with a uniform means of communicating with various host computer networks, irrespective of the communication protocols of the wired

communication network 10. All protocol and other feature-specific communications translation and handling is performed at the wired communication network interface module 100. The wired communication network interface module 100 may be designed for networks utilizing different implementations, such as transparent asynchronous communication, Hayes compatible communication, TCP/IP stream socket, Bidirectional Messaging Facilities, File Transfer Facilities, SNA Protocol Enveloping, Vehicle Location Reporting Facilities, Credit Card Verification Facilities, and Harris DNP 3.0 Frame Relay. As noted above, a unique set of procedures may be provided by the wired communication network interface module 100 for each type of wired communication network 10. Examples of several of these sets of procedures are provided below.

Referring now to Fig. 20, there is illustrated an exemplary flow chart of the set of procedures associated with the wired communication network interface module 100 that are designed for transparent asynchronous communication networks. At step 900, an initialization process begins in accordance, for example, with step 802 of Fig. 15. At step 902, the serial port of the wired communication network 10 that is to be used is determined, and the identified port is opened or accessed at step 904. At step 906, the speed (bps), the parity (odd or even), and the data bits for communication with the network 10 are established along with other appropriate parameters.

At step 908 a data read operation begins. The data read routine may be invoked by the outbound data event handler 96 at step 850 (see Fig. 18). Initially, at step 910, the wired communication network interface module 100 determines if there is data to be read from the serial port. If at step 910 the wired communication network interface module 100 determines there is data to be read, then at step 916 the

data is added to an accumulated data buffer within the remote network controller 20. At step 918, if the wired communication network interface module 100 determines that the data buffer is full, then at step 914 the data accumulated in the buffer is returned to the calling module. If, however, at step 910 the wired communication network interface module 100 determines there is no data to be read from the serial port attached to the network 10, then at step 912 it is determined if the inter-character time (e.g., a predetermined character receipt delay time) has been exceeded. If at step 912 the inter-character time has been exceeded, then at step 914 the accumulated data buffer is returned to the calling module to be further processed. Otherwise, if the inter-character time has not been exceeded, then processing continues at step 910 so that the wired communication network interface module 100 may again determine if there is data to be read from the serial port.

At step 920, a write data routine is started. The write data routine may be initiated by the inbound data event handler 94 at step 836 in Fig. 17. At step 922, the data is sent directly to the serial port of the wired communication network 10. The write data routine is thereafter completed.

At step 924, a terminate routine is initiated. The terminate routine may be initiated in accordance with a terminate request at step 818 in Fig. 15. In response to initiation of the terminate routine, the serial port of the network 10 is closed at step 926. Thereafter, at step 928, the serial port resource is released so it may be used by another process. Finally, at step 930, any data buffers in use are also released.

Referring now to Fig. 21, there is illustrated exemplary flow charts of the set of procedures associated with the wired communication network interface module 100 for networks utilizing TCP/IP stream socket connectivity. At step 932, an

initialization process begins in accordance, for example, with step 802 of Fig. 15. At step 934, a host network and serial port to be used are determined. At step 936, an appropriate socket is created. At step 938, the socket server is accessed for data transport.

At step 940 a data read operation begins. The data read routine may be invoked the outbound data event handler 96 at step 850 (see Fig. 18). Initially, at step 942, the wired communication network interface module 100 determines if there is data to be read from the socket. If at step 942 the wired communication network interface module 100 determines there is data to be read, then at step 944 the data buffer is returned to the remote network controller 20 for further processing.

At step 946, a write data routine is started. The write data routine may be initiated by the inbound data event handler 94 at step 836 in Fig. 17. At step 948, the data is sent by the wired network interface module 100 directly to the socket at the wired communication network 10. The write data routine is thereafter completed.

At step 950, a terminate routine is initiated. The terminate routine may be initiated in accordance with a terminate request at step 818 in Fig. 15. Initially, the socket is closed at step 952. Thereafter, at step 954, the socket resource is released so it may be used by another process. Finally, at step 956, any data buffers in use are also released.

Referring now to Fig. 22, there is illustrated exemplary flow charts of the set of procedures associated with the wired communication network interface module 100 for use with networks utilizing Vehicle Location Reporting Facilities. At step 958, an initialization process begins in accordance, for example, with step 802 of Fig. 15. At step 960 the wired communication network interface module 100 determines a

position recording file to be used to record data. The position recording file may be stored within the wired communication network 10. At step 962, the position recording file is accessed from the network 10. Thereafter, at step 964, the recording interval is placed into a read buffer that may be provided to the remote device 52 via the mobile data controller 54.

At step 966 a data read operation begins. The data read routine may be invoked by the outbound data event handler 96 at step 850 (see Fig. 18). Initially, at step 968, the wired communication network interface module 100 determines if there is data in the read buffer. If at step 968 the wired communication network interface module 100 determines there is data in the read buffer, then at step 970 the contents of the read buffer is returned to the calling module.

At step 972, a write data operation is started. The write data routine may be initiated by the inbound data event handler 94 at step 836 in Fig. 17. Initially, at step 974, the wired communication network interface module 100 determines if a full Global Positioning Satellite (GPS) message has been accumulated. If at step 974, the wired communication network interface module 100 determines that a full GPS message has not been accumulated, then no action is taken at step 980. If at step 974, the wired communication network interface module 100 determines that a full GPS message has been accumulated, then the message is converted at step 976 to a standard form usable by the wired communications network 10. At step 978, the position recording file at the host is appended with the GPS message.

At step 982, a terminate routine is initiated. The terminate routine may be initiated in accordance with a terminate request at step 818 in Fig. 15. The terminate routine may comprise closing the position recording file at step 984.

Referring now to Figs. 23A and 23B, there is illustrated exemplary flow charts of the set of procedures associated with the wired communication network interface module 100 that are designed for networks utilizing Bidirectional Messaging Facilities or Store and Forward Messaging Facilities. Referring to Fig. 23A, at step 986, an initialization process begins. The initialization process may be started in accordance, for example, with step 802 of Fig. 15. At step 988, a message queue for the remote device 52 is accessed. Thereafter, at step 990, if no queue exists for the remote device 52, a message queue is created at the remote network controller 20.

At step 992 a data read operation begins. The data read routine may be invoked by the outbound data event handler 96 at step 850 (see Fig. 18). Initially, at step 994, the wired communication network interface module 100 determines if a message is in the process of being sent. If at step 994 the wired communication network interface module 100 determines that a message is being sent, then at step 998, the current message segment is sent by the remote device 52 to the wired communication network 10. If at step 994 the wired communication network interface module 100 determines that no message is being sent, then at step 996, the wired communication network interface module 100 determines if there is a message queued. If at step 996 the wired communication network interface module 100 determines that no messages are queued then no action is taken. However, if at step 996 the wired communication network interface module 100 determines that there is a message queued for the remote device 52, then the current message queued is sent at step 998. After the last segment of the message is sent, the wired communication network interface module 100 may indicate that delivery of the message is pending to the remote device 52 at step 1000.

At step 1002, a write data routine is initiated. The write data routine may be initiated by the inbound data event handler 94 at step 836 in Fig. 17. Initially, at step 1004, the wired communication network interface module 100 determines if a new message is present. If at step 1004 the wired communication network interface module 100 determines that a new message is present, then at step 1010 the wired communication network interface module 100 starts a new queue entry. Thereafter, processing continues at step 1006, where the message segment is recorded. At step 1008, the wired communication network interface module 100 determines if this is the last segment. If at step 1008 the wired communication network interface module 100 determines that it is the last segment, then at step 1012, the queue entry is closed and "Message Received" message may be placed into the read buffer at step 1014 to be sent to the remote device 52. If at step 1008, the wired communication network interface module 100 determines that it is not the last segment, then no action is taken.

Referring to Fig. 23B, at step 1016, a terminate routine is started. The terminate routine may be initiated in accordance with a terminate request at step 818 in Fig. 15. At step 1018, the wired communication network interface module 100 determines if the terminate request has come in the middle of a message. If at step 1018, the wired communication network interface module 100 determines that the request came in the middle of a message, then the message is purged at step 1022. Processing then continues at step 1020, where the wired communication network interface module 100 closes the message queue.

Referring now to Fig. 24, there is illustrated flow charts of the set of procedures associated with the wired communication network interface module 100 designed for use with networks utilizing Credit Card Point-of-Sale Facilities. At step

1024, an initialization process begins. The initialization process may be started in accordance, for example, with step 802 of Fig. 15. In accordance with the invention, no action is required for the initialization procedure.

At step 1026 a data read operation begins. The data read routine may be invoked the outbound data event handler 96 at step 850 (see Fig. 18). Initially, at step 1028, the wired communication network interface module 100 determines if there is a response from a server attached to the wired communication network 10. If at step 1028 the wired communication network interface module 100 determines there is a response from the server, then at step 1030 the response is read, and the response buffer is returned to the remote device 52 at step 1032. If at step 1028 there is no response from the server, then no action is taken.

At step 1034, a write data operation is started. The write data routine may be initiated by the inbound data event handler 94 at step 836 in Fig. 17. At step 1036, the wired communication network interface module 100 first determines if a full request by the remote device 52 has been accumulated. If at step 1036 the wired communication network interface module 100 determines that a full request has not been accumulated, then the remote device 52 continues to accumulate a request at step 1042. If at step 1036 a full request has been accumulated, then at step 1038, the request is formatted for the server on the wired communications network 10. Thereafter, the request is placed into the server file at step 1040.

At step 1044, a terminate routine is illustrated. The terminate routine may be initiated in accordance with a terminate request at step 818 in Fig. 15. According to the present invention, no action is necessary for the terminate routine.

Referring now to Fig. 25, there is illustrated a block diagram of the various components of the mobile data controller 54, according to another aspect of the present invention. The mobile data controller 54 may be specifically designed to match the asynchronous data transferred to and from the remote device 52 to the radio infrastructure 56 protocol. Typically, there is one type of mobile data controller 54 that is associated with a host data controller 22 which is connected to the mobile interface 24 of the remote network controller 20. In addition, the mobile data controller 54 may have a unique identifier associated with it for routing purposes.

In accordance with the present invention, the mobile data controller 54 may be implemented by any combination of hardware and software. For example, the mobile data controller 54 may comprise a commercially available processor with overlying software and random access memory. The software running in the mobile data controller 54 may be written in Z80 or other appropriate processor-based (i.e., native) assembly language and configured to the specific radio infrastructure 56. The software may specify the various voltage levels and logic signals necessary to communicate via the RF communications infrastructure 56. As noted above, the mobile data controller 54 may translate and pass any protocols associated with the wired communications network 10 to and from the remote device 52 to make it appear to the wired communication network 10 that the remote device 52 is locally-attached.

The mobile data controller 54 is configurable over the radio infrastructure 56. configuration information may be input by an operator at the remote network controller 20 through the console interface 34 (see Fig. 2) and passed over the radio infrastructure 56 to the mobile data controller 54. This allows parameters such as

packet size to be changed at the host wired communications network 10 without the necessity of altering the mobile data controller 52 in the field.

As shown in Fig. 25, the mobile data controller 54 may comprise an RF communications interface module 106, a remote device communications interface module 102, and a configuration and monitoring module 104. The mobile data controller 54 may be connected to the remote device 52 via a communication port 108 for sending and receiving data. The remote device communications interface module 102 is connected to the remote device 52 via the communication port 108 and is responsible for sending data to, and receiving data from, the remote device 52. The communication port 108 may comprise, for example, an RS-232 adapter.

The configuration and monitoring module 104 is specific to the type of radio infrastructure 56 employed. Software parameters, such as the number of subsystem ports, how often to send health and status requests, and a list of host data controllers 22 to which the mobile data controller 54 can communicate, may be set and stored in the configuration and monitoring module 104. The configuration and monitoring module 104 can also accumulate statistics which are passed to the host data controller 22.

In order to diagnose potential system errors in the mobile data controller 54, an operator may be provided with the ability to field test and analyze the mobile data controller via an external diagnostic port 112 to determine a cause of the system error or failure. The diagnostic port 112 may be used not only to determine if the mobile data controller 54 is operational, but also can be used to configure software parameters to determine the type of the radio infrastructure 56. These parameters can

be changed to communicate with a different type of radio infrastructure 56 as necessary.

The RF communications interface module 106 is responsible for sending and receiving the data via radio-frequency (RF) transmission. The RF communications interface module 106 is specific to the radio infrastructure 56 used, and is connected to the radio infrastructure 56 through a communication line 110. Because the mobile data controller 54 is designed to integrate with an existing radio infrastructure 56, each mobile data controller 54 may be software configured, for purposes of flexibility, to work with many types of radio infrastructure 56 protocols. The RF communications interface module 106 may also send health and status information regarding the mobile data controller 54 to the host data controller 22. This information may inform the remote network controller 20 that the mobile data controller 54 is operational and the remote device 52 is accepting data.

According to the present invention, the RF communication interface module 106 may include a commercially available modem (not shown). The modem may be selected depending on the data rate(s) of the communication line 100 and the radio infrastructure 56. More than one modem may be provided if multiple data rates are required. Optionally, the modem can be implemented using a Digital Signal Processing (DSP) chip and associated software. The DSP chip can be a commercially available programmable signal processing chip. In such a case, the DSP implementation will allow a single modem to be changed (e.g., by uploading new parameters to the DSP software) in order to communicate with a plurality of different types of radio infrastructures 56 having distinct protocols and data rates.

Similar to the host data controllers 22, the mobile data controllers 54 may be compatible with, for example, conventional point-to-point radio systems, conventional repeater-based radio systems, LTR Trunking, Motorola Trunking, Ericsson (EDACS) Trunking-Voice Path, EDACS RDI Trunking-Data Path, and EDACS IMC Voice Path based radio infrastructures 56.

Referring again to Fig. 2, a brief description of the interprocess communications manager 28 will be provided. According to the present invention, the interprocess communications manager 28 is responsible for routing all communication between the various modules and interfaces within the remote network controller 20. The interprocess communications manager 28 creates a logical route from the remote device 52 to the service interface 30 of the remote network controller. The interprocess communications manager 28 passes routing information which determines from which radio infrastructure 56 and remote device 52 the inbound data has come from, and to which radio infrastructure 56 and remote device 52 the outbound data will be sent.

The interprocess communications manager 28 may also pass information generated by the remote network controller 20, which is independent of the data and routing information. This information may include internal parameters and error detection codes. The interprocess communications manager 28 also interfaces with the control process module 26. The control process 26 may act as the "central hub" of the remote network controller 20. The control process 26 provides resource management, process management, session management, configuration management and system statistics management within the remote network controller 20.

As further shown in Fig. 2, the remote network controller 20 also includes a console interface 34. The console interface 34 may be adapted to allow a network operator to configure and control the wireless Network Interfaces described above, the mobile user characteristics and the configuration information of the wired communications network 10. The console interface 34 may be a stand-alone platform having a commercially available processor (e.g., an Intel or Motorola based processor) and an Ethernet controller card for communicating, for example, with another remote network controller 20.

Referring now to Fig. 26, there is illustrated a remote gateway 120, in accordance with another aspect of the present invention. As shown in Fig. 26, the remote gateway 120 is comprised of a transparent communications module 122, a field service interface module 128, a configuration and health module 124 and a RF communications module 126. The remote gateway 120 is functionally similar to the mobile data controller 54. However, the remote gateway 120 is a specific type of mobile data controller that may be used to attach to a remote telemetry unit for monitoring, for example, electrical power distribution.

The transparent communications module 122 is responsible for communicating with a terminal device, typically a Remote Telemetry Unit (RTU) located in the field, and accepts data from the RF communications module 126. As shown in Fig. 26, the transparent communications module 122 may be connected to a RTU 133 via a communication line 130. The transparent communications module 122 does not recognize any protocol, but handles hardware flow control and buffering and packetizing. Data communication between the transparent communications module 122 and the RTU 133 may be carried through asynchronous serial transfer.

The RF communications module 126 is configured to communicate with the remote network controller 20 using whatever protocol is required for data transport over the radio infrastructure 56. The RF communications interface module 126 interfaces with the radio infrastructure 56 in a similar manner as previously described above with regard to the RF communications interface module 106. The RF communications interface module 126 accepts data from the transparent communications module 122 and delivers it to the radio infrastructure 56 for transmission to the remote network controller. The RF communications interface module 126 detects collisions with inbound RF data and restarts outbound transmissions. The RF communications interface module 126 performs error/retry functions and notifies the transparent communications module 122 of successes or failures.

The field service interface module 128 allows a technician to field test the remote gateway 120 and troubleshoot the remote gateway should a system error or problem arise. An external diagnostic port 112 connected to the field service interface module 128 may be provided for this purpose. The field service interface module 128 may interact with the configuration and health module 124 to query, set, and reset local configuration of the remote gateway 120.

The configuration and health module 124 may accept configuration information from the remote network controller 20 via the radio infrastructure 56 and adjust the operating parameters of the remote gateway 120 accordingly. The configuration and health module 124 may also monitor and determine if the RF communications module 126 has successfully transmitted a packet of information to the host data controller 22 by analyzing the data stream. If a packet of information

has not been successfully transmitted, the configuration and health module 124 may direct the RF communications module 126 to resend the packet of information to the host data controller 22.

Referring now to Figs. 27 and 28, there is illustrated a block diagram of an integrated remote network controller 140 according to another aspect of the present invention. The components of the remote network controller 140 that are similar to that discussed above with respect to Fig. 2 are designated with the same reference numeral and also with a prime (" ' ") added thereto.

As shown in Fig. 27, the remote network controller 140 according to another aspect of the present invention may include one or more service interfaces 30', an interprocess communications manager 28', a control process 26', one or more mobile interfaces 24', and a subsystem synchronization processor 150 which is used to link one or more remote network controllers 140 together (see Fig. 28). As further shown in Fig. 27, one or more host data controllers 22' are connected to the mobile interfaces 24' and provided externally to the remote network controller 140. The number of host data controllers 22' and mobile interfaces 24' may be dependent on the number of radio infrastructures 56 present. In addition, the number of service interfaces 30' may be dependent on the number and type of wired communications networks 10 present.

According to an aspect of the present invention, two remote network controllers 140 may be linked by a local network 152 (see Fig. 28). This configuration provides a redundant system which insures greater reliability of communication between the remote device 52 and the wired communications network 10. For example, should any particular component fail, such as a host controller 22' or an interprocess communications manager 28', the remote device 52 can still

communicate with the wired communication network 10 because of the redundancy of the components of the remote network controllers 140 provided.

There are two main implementations of this system according to the present invention. With the first implementation, only one of the remote network controllers 140 is operating at any given time. Should the operational remote network controller 140 fail, the other remote network controller 140 may immediately take its place. For example, if remote network controller "A" fails, then remote network controller "B" may be activated to take its place.

Under the second implementation, a distributed processing scheme is utilized and both of the remote network controllers 140 are operated at the same time. According to the distributed processing scheme, the processing load (e.g., event handling and data transfer) may be distributed among the operational remote network controllers 140. Should a particular remote network controller 140 (e.g., controller "B") fail, the remaining operational remote network controller 140 (e.g., controller "A") will handle the entire processing load.

The above-mentioned distribution of the processing load is generally a function of the radio infrastructure 56, and not the processing capacity of the remote network controllers 140. This is because performance increases are mainly based on the number of available communications channels, rather than the raw processing capability of the remote network controllers 140 attached to the wired communications network 10. For example, if the radio infrastructure 56 is a trunking radio network with five channels, it is possible for all five channels to be simultaneously allocated and used by the remote network controllers 140. On the other hand, if there is only one channel available in the radio infrastructure 56, then

only one remote network controller 140 can access the radio infrastructure 56 at a time; as a result, no performance gain may be realized by employing multiple remote network controllers 140 when only limited channels are available.

As illustrated in Fig. 28, the local network 152 is used to connect the remote network controllers 140 together. The local network 152 may be, for example, an Ethernet local area network. Each of the remote network controllers 140 includes a subsystem synchronization process module 150 that is connected to the local network 152. Two separate console interfaces 34' may also be attached to the local network 152. Each console interface 34' may be attached to the local network 152 to allow an operator to configure and control a particular remote network controller 140.

The subsystem synchronization process module 150 may be implemented through any combination of hardware and software and is responsible for keeping track of all routing tables and health and status information associated with both of the remote network controllers 140. Each subsystem synchronization process module 150 is connected to an interprocess communications manager 28' of one of the remote network controllers 140 and may access all routing tables and health and status information with respect to the remote network controller from the interprocess communications manager. The health and status information and routing tables may be periodically updated based on the status of and events present at the remote network controller 140. The periodically updated health and status information and routing tables may then be shared with the other subsystem synchronization process module 150 via the local network 152 so that the tables and information associated with both of the remote network controllers 140 is maintained in each of the subsystem synchronization process modules 150. Since the tables and information are

periodically updated, a synchronization routine may be provided so that the information and tables are sent to the respective subsystem synchronization process modules 150 at predetermined intervals. If a particular subsystem synchronization process module 150 does not send or receive the tables and information, or if a particular subsystem synchronization process module 150 sends information indicating that one of the remote network controllers 140 has malfunctioned, the other subsystem synchronization process module 150 may reroute any existing connections to the host data controllers 22' and to the wired network 10 of the malfunctioning remote network controller 140 to the remaining operational remote network controller 140.

As further shown in Fig. 28, the host data controllers 22' have two ports, 154 and 156, that are connected to a different remote network controller 140. As in Fig. 2, the remote network controller 140 communicates to the host data controller 22' and sends health and status information through the ports. If the host data controller 22' does not receive information that one of the remote network controllers 140 is operational, the host controller 22' can switch ports, e.g., from port 154 to port 156, in order to communicate with the other remote network controller 140.

While the invention has been described with references to several exemplary embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitation. Changes may be made, within the purview of the appended claims, without departing from the scope and spirit of the invention in its aspects.

For example, although Fig. 28 only shows two remote network controllers 140 that are connected by a local network 152, it is possible to connect two or more

remote network controllers by the local network 152 to provide increased redundancy. In addition, a plurality of local networks 152 may be provided to connect the remote network controllers. Other modifications to the present invention may include selectively processing inbound and outbound data in a different logic order and/or by different components. In accordance with such a modification, processing functions may be performed only by the control process, or in the interprocess communication manager. Another application may be combining the mobile interface with a host data controller, and placing the integrated unit within the remote network controller.

Referring now to Fig. 29, therein is illustrated a general overview of another embodiment of the present invention which includes a mobile Router 200 in accordance with an aspect of the present invention. The Router 200 provides the mobile application or device 52 with the capability to selectively transmit and receive data over a plurality of radio frequency infrastructures 56 and/or the public switched telephone network 58 in accordance with user configured parameters.

Referring now to Fig. 30, therein is illustrated a schematic block diagram of the mobile Router 200. In the following description of the Router 200, each of the elements will be initially generally described and in greater detail thereafter. As shown in Fig. 30, the mobile application or device 52 may be attached to multiple Networks by the Router 200 through Network Interfaces 214A-D, a Router Core 204, and a Switch 212. The Network Interfaces 214A-D provide connectivity for data between the Switch 212 and the various Networks infrastructures (e.g., radio infrastructures 56 and public switch telephone network 58) through which the mobile device or application 52 connects to the communications network 10 (see Fig. 1). The Switch 212 is actuated by the Router Core 204, and sends data to a fixed host

application or device (e.g., RNC 20) via the selected network. The Network Interface 214 provides information to the Network Availability process 210, which sends this information to the Decision process 206. The Decision process 206 operates in accordance with User Configured parameters 208 which specify when and through which Network the data is to be transmitted. The decision process 206 monitors the User Configuration parameters 208, and the Network Availability 210. When the Decision process 206 (in accordance with User Configuration 208 parameters) specifies that a Network (e.g., Network 3) different than the Network currently in use (e.g., Network 1) should be used, the Decision process 206 checks the Network Availability 210 for the particular Network to be switched to. If the Network is available, the Decision process 206 instructs the Router Core 204 to switch to the new Network. The Router Core 204 then updates routing tables (not shown) maintained within the Router Core 204 to reflect the new data path, and actuates the Switch 212 to connect to the new Network. Data may then flow over the new Network. In accordance with an aspect of the present invention, data may flow inbound to the fixed host through one Network, and outbound to the remote mobile Application or device 52 through the same Network, or through a different Network.

With reference to Fig. 30, the mobile application or device 52 may comprise a software application running on a portable or laptop computer performing a variety of functions as programmed by the software application (e.g., database services). The Application or device 52 may also comprise a special purpose device designed to perform a particular function, such as a credit card reader or barcode scanner. The Application or device 52 may generate a data stream which is sent to a fixed location (e.g., a host computer infrastructure 10).

An exemplary application running on the mobile device 52 is a mobile remote client application which provides the remote user with the capability to send and retrieve data from a fixed database server application. The data may consist of customer records which, for example, may be used by service personnel operating a fleet of vehicles to service customers scattered about a wide geographic area. In the exemplary application, the mobile client application may request customer records from the fixed database server, and display the records for viewing by mobile service personnel. The mobile client application may send updated records to the fixed database as the service personnel finish assigned tasks. The updated records may contain a service history, equipment upgrades, and repairs for each customer.

Another exemplary application running on the mobile device 52 may be a client application which retrieves a list of dispatched jobs to be performed by the service personnel during each day. The jobs may be uploaded to the remote mobile device 52 each morning and stored in another client application in the mobile device 52. As the service personnel change job locations, the status of each job may be updated to indicate a status, e.g., en route, arrived and finished with comments. The status may be sent from the application to the fixed home office, so a dispatcher at the home office is aware of the locations of service personnel in the field.

By way of non-limiting examples, the mobile device 52 may comprise a portable or laptop computer; a computer having an embedded Router 200; a terminal or terminal emulator; a data gathering device (e.g., a SCADA system or remote telemetry system for obtaining data from a remote location for forwarding to a central location for processing); a card-swipe reader device (e.g., credit/debit/bank cards) for use in a mobile billing application, such as a taxi or mobile food cart; a smart-card

reader; a logging device, such as those used in a package delivery system or fleet; a device for reading bar codes (e.g., for inventory control); and a remote application with data to send or to receive, from a fixed application or device (e.g., remote diagnostic tool). The above-noted applications are provided merely for exemplary purpose, and other applications and mobile devices 52 may be used with the Router 200 of the present invention.

Typically the device or Application 52 sends and receives data using a variety of protocols (e.g., Internet Protocol (IP) / transparent (via MDC 54) / ack-nack, etc.). The use of a variety of protocols provides for open transport of data throughout many networks, and in particular, networks which support open standards such as IP. However, many proprietary networks which require interface and/or protocol translation remain in use. In the Router 200 of the present embodiment, the function of interfacing with networks and protocol translation may be performed by the Network Interfaces 214A-D.

According to another aspect of the invention, other types of devices may be connected to the Network Interface 214. Such devices may be used for functions other than data and voice communication. By way of non-limiting examples, these devices may include Global Positioning System (GPS) receivers and application processors.

The Router Core 204 is a function which shuttles messages between the Application or Device 52 and the various Networks. In accordance with the present embodiment, the router Core 204 may control which network of a plurality of usable network messages are to travel over, and connect access ports (described below) to each Network and the Application or Device 52.

The Router Core 204 may also comprise a list of all possible "names" or "addresses" to which data may be sent, or from which data may be received. The local "names" or "addresses" of the Router Core 204 are stored in tables within a memory (not shown) of the Router Core 204. Thus, the Router Core 204 may serve as a communications "address book" for the Router 200 of the present embodiment. The Router Core 204 also checks all messages passing through, and decides, based on the address and/or name entries in the tables, if the message is relevant to the attached Application or Device 52, or to the fixed host (e.g., RNC 20). The address of the fixed host may be stored in the Router Core table as well. In accordance with the table entries, received messages may be determined to be valid or invalid. The Router Core 204 may also actuate the Switch 212 in accordance with the output of the decision process 206. The Switch 212 is actuated such that incoming and outgoing messages can be sent through the "current" network, as determined by the decision function 206 (to be described below).

The Switch 212 may comprise a message multiplexor, i.e., the Switch 212 performs a one-to-many function for in-bound messages (to the fixed hosts), and a "many-to-one" function for outbound messages (from the fixed host). As noted above, the appropriate network selection is made by the Router Core 204 in accordance with the output of the decision process 206. Messages travel through the Switch 212, the Router Core 204, and the current Network Interface 214.

Referring to Fig. 32, the Switch 212 may be implemented using a combination of hardware (e.g., multiple electronic ports, one per Network Interface 214) to perform the physical connection process 212B, and software (e.g., handlers which are interrupted at each character to move the character to either the Router Core 204

(outbound), or to the current Network Interface 214 (in-bound)) to perform the switch logic process 212A.

As a non-limiting exemplary hardware implementation, the Switch 212 may comprise an 80386EX microprocessor, running at 33 MHZ, 256 kilobytes of FLASH ROM, 512 kilobytes of static RAM, six asynchronous serial ports, two TTL-to-RS232 convertors interfacing with two of the six serial ports directly to compatible devices external to the Switch 212, and four internal TTL serial interfaces to internally-mounted daughter boards, which carry Network Interfaces 214A-D. Each Network Interface 214 mounted on a daughter board may include a power supply for the Network Interface, a serial interface to the 80386EX microprocessor, and an interface to the outside network. The outside network may be a radio, a LAN, an antenna (for internally-mounted radios in the Network Interface 214), or other device accepting or supplying data from/to the Router 200.

The Switching function of the Switch 212 is provided by each serial Network Interface port at the 80386EX microprocessor, and the software residing in FLASH ROM. The software logic determines which Network Interface to use for transmission and receipt of data. The decision is implemented in the Switch, by selecting a physical serial port (and therefore which Network Interface) is to be used as the current Network. The Decision software in the FLASH ROM instructs the microprocessor to send the data to a specific serial port (which is mapped to specific physical addresses within the address range of the 80386EX microprocessor). The microprocessor then constructs the next message in the message buffers in RAM, and sends the message through the specific serial port which is designated as the "current Network" Interface port. The data then goes to the Network Interface (e.g., network

interface 214A) connected to that specific serial port and on to the Network infrastructure. Received data is input to the Network Interface (e.g., network interface 214B which may be set to the "current Network" serial port) and the microprocessor, where the received data is processed by the microprocessor. In accordance with an aspect of the present invention, messages which are received through Network Interfaces which are not designated as the "current Network" are ignored.

The Network Interfaces 214A-D are devices which present data to, or obtain data from the radio operating at the various R.F. Network frequencies, bandwidths, and modulations. The Network Interfaces 214A-D may provide a port through which messages pass, to and from the Switch 212. The messages are typically in the form of a sequence of serial digital bits. The Network Interfaces 214A-D also may provide a modulator/demodulator function which transforms the digital data into an analog form which may be easily sent through the R.F. Network voicepath, based on characteristics of the assigned frequency band of the R.F. Network. The characteristics of analog transmissions are typically bandwidth (in Hertz, or cycles per second), noise present in the Network, and assigned frequency of the R.F. Network. Further, the Network Interfaces may interface with a radio, which may be included within the Network Interface 214, or may be mounted externally to the Router 200 (as shown in Fig. 29). The Network interface 214 radio interface comprises the actual physical connections to the radio for the voicepath data, the muting function (if present and/or required) and the functionality for issuing a Press-to-Talk to the radio, and for receiving the Transmit Grant signal from the radio; both are used for handshaking between the radio network and the Network Interface 214. This

handshaking may be necessary for proper timing of the data out onto the RF Network. The muting function is used for silencing received signals which represent data, rather than voice traffic, which enables a remote user to mute the audible noise of the data traffic, which can be annoying to the remote user.

Examples of Network Interface 214A-D include the MDC 54 and the NovaTel Wireless NRM-6812 Cellular Digital Packet Data (CDPD) modem. Where the network interface 214 comprises the MDC 54, the radio is mounted external to the MDC 54, whereas in the NovaTel example, the radio and the network interface are integrated into a single unit.

As noted above, the Network Interfaces 214 provide connections to various types of networks. These networks may be wired (for example Public Switched Telephone Network 58), or wireless (for example Cellular Digital Packet Data (CDPD)). The following non-limiting list includes networks that may be interfaced to the Router 200 by the Network Interfaces 214A-D: private voice radio including conventional and trunked radios (e.g., using MDC 54), Cellular Digital Packet Data (CDPD), Spread Spectrum (e.g., direct sequence and channel-hop), GSM, GPS receiver, satellite transponder, RDI (Ericsson) interface, AMPS, RAM Mobile (Mobitex), RS232, RS485, Angel (AT&T), Asynchronous Transfer Method (ATM), Integrated Services Digital Network (ISDN), public switched telephone network (PSTN (POTS) telephone network), Ethernet, Ardis, Personal Communications Services (PCS), and any other network which is either transparent or operates using a specific protocol.

The specific protocols to the above-listed networks are implemented in the Network Interfaces 214A-D. These protocols may be very different, and therefore

incompatible with each other. Additionally, a translation device may be provided in each Network Interface 214 to translate between IP and the particular network protocol. By providing such a translation device, the Application or Device 52 can use IP data regardless of the particular network the Application or Device 52 is actually using.

Referring to Fig. 31, a description of the functional components of the Router 200 will now be described. The Router 20 may be implemented as an autonomous device with multiple connections to the networks through which data is to be routed. The user Configuration Interface 208 provides a means whereby an external device such as a keyboard/terminal may be used to supply configuration information such as preferred routes, network node addresses, etc. to the router. Such information is accepted by the Configuration Interface 208 and is placed into a non-volatile store (e.g., memory) which may be queried by other router components. In addition, capability may be provided whereby diagnostic information may be requested from the router and sent to the terminal device for evaluation by a technician.

The Router Core 204 is responsible for making all routing decisions. For a given destination network address specified within a data packet or datagram received from one of the network interface drivers 215A-D, the most-preferred path will be selected and the data packet or datagram forwarded through the preferred network interface driver 215A-D. Routing decisions are generally based upon such metrics as network speed and interface availability. Other metrics such as destination network, time of day, type of data, etc. may also be incorporated into the routing decision. Further, routing decisions may be made at the packet level such that each individual

packet of data may be transmitted and/or received on different networks in accordance with the user configured parameters 208.

Exemplary Network Drivers 215A-D may include an Ethernet Driver, a Token-Ring Driver, and a Serial PPP Driver. The Ethernet Driver provides a means for sending and receiving data through an Ethernet-type network. The function of the driver is to shield the Router Core from the details of network media access. The Token-Ring Driver provides a means for sending and receiving data through a Token-Ring-type network. The function of the driver is to shield the Router Core from the details of network media access. The Serial PPP Driver provides a means for sending and receiving data through a PPP-based serial data link. The function of the driver is to shield the Router Core from the details of network media access. Other drivers 215A-D may be provided to interface with other types of networks as necessary.

The Network Availability 210 (see also Fig. 30) is a function which periodically interrogates each installed Network Interface 214 in the Router 200 and may determine if the Network Interface 214 is installed; if the Network Interface 214 is properly configured and functioning properly; if the Network Interface 214 is connected to the Network, on-line, and available for sending/receiving messages; and if the Network Interface 214 is in good health. The above interrogation process may be accomplished by monitoring a timer tick (provided by the switch microprocessor), which instructs the Network Availability 210 to query each Network Interface 214. When the timer tick occurs, the Network Availability 210 function interrogates each Network Interface 214 as noted above. The status of each Network Interface 214 is then passed to the Decision process 206, which determines what the "next Network"

will be if the result of the interrogation indicates that the "current Network" is experiencing transmission problems.

The Network Availability 210 of each Network Interface 214 is determined in a manner specific to the particular interfaced Network. For example, if the Network is CDPD, the Network Availability 210 interrogates the network to determine if the Network Interface 214 is currently registered with the Network, and therefore active. Also, in the CDPD network, the Network Availability 214 determines if the Received Signal Strength Indication (RSSI) is sufficient to transmit relatively error-free data. For example, in the NovaTel CDPD Network Interface a RSSI of -100 dBm will provide for good data transmission qualities. Thus, if the Network Availability 210 function queries the NovaTel CDPD Network Interface for the RSSI, and the response is -110 dBm, then the signal is too weak for error-free transmission, and therefore cannot be used at this time. This information is passed to the Decision process 206 to determine if the "current Network" should remain the "current Network", and if not, to determine what the "next Network" should be.

The User Configuration 208 block is used to define user configurable parameters by which the Router Core 204 selects the "current Network" and the "next Network". The Router parameters may include the date and time (e.g., yr-mo-da, hh:mm:ss), and the Network Interface 214 installed in each of the internal slots of the Router 200. According to the present embodiment there are six internal slots to accommodate Network Interfaces to any of private voice radio using e.g., the MDC 54 and a variety of radios, both conventional and trunked; Cellular Digital Packet Data (CDPD), such as Sierra Wireless or NovaTel CDPD modems; spread spectrum, either direct sequence, or channel-hop, as Xetron Hummingbird spread spectrum

modem; GSM, such as Ericsson serial GSM module; GPS receiver, such as Motorola VP Encore GPS receiver, or Trimble SVEE Six receiver; satellite transponder; RDI (e.g., Ericsson) interface, implemented via a software protocol module and quasi-RS232 interface to radio; AMPS; RAM Mobile (e.g., Mobitex); RS232 default and fixed for example in slots 1 and 2; RS485; Angel (e.g., AT&T); ATM; ISDN; PSTN; Ethernet; Ardis; PCS; any other network which is either transparent or operates using a specific protocol; and none. Although six slots are disclosed herein, other numbers of slots may be provided.

Other user configurable parameters include: the priority of each internal slot, (e.g., 1 to 6) where the slot with priority 1 is the default startup slot and Network; baud rate of each slot (a default rate may be set to 9600 bits per second, but may be configured to be any standard baud rate, divisible by 300, up to 115.2 kilo bits per second); cost per kilobyte per slot (e.g., $0.xx per kilobyte where the least costly slot that is available and highest priority will be default); protocol per slot (e.g., none, Point to Point (PPP), Serial Line Internet Protocol (SLIP), Hayes "AT" commands, transparent); slot mode, for example, transparent, PSTN, cellular, IP, receive only; slot name or address or phone number; slot to be used for diagnostics (e.g., default may be set to slot 2); slot muting to be used (e.g., none, PL, DTMF, other); number of retry transmissions per Network Interface (per slot) before declaration of Network Interface failure (e.g, 0-10); if slot Network Interface needs to be configured before it can operate (e.g., y,n); slot to be used for remote display (e.g., default may be set to slot 2); slot to be used for Device or Application 52 (e.g., a connection to a mobile computer; default is slot 1); and frequency at which Network Availability 210 is

checked (e.g., default may be set to five seconds). Other user configurable parameters may be introduced and configured as necessary.

The User Configuration 208 function provides the user with the capability to instruct the Router 200 how to select a particular Network. These metrics may include, but are not limited to: which Network is connected to which Router port, time of day and date, priority (switching sequence) of each Network, cost per packet of each Network, and preferred default Network.

On power up, the User Configuration 208 is checked to determine if it is current. If the User Configuration 208 is determined to be out of date, the end user is requested to input a configuration. The user is notified by blinking LEDs on the front panel or by a message sent to the mobile device 52. If the User Configuration 208 is determined to be current, no user input is requested.

Further, each Network is continuously evaluated for health and connectivity status. There are a number of parameters which are examined to determine this, including, but not limited to: Received Signal Strength Indication (RSSI), Clear to Send (CTS), Channel Clear/Channel Ready, and Transmit Grant.

The Decision process 206 continuously examines the User Configured parameters in the user configuration block 208, to determine the next Network to use, in case the current Network becomes unavailable to send or receive data. Such an unavailability may arise because the remote user (and consequently the Router 200) has moved beyond coverage of the Network, or because a problem has occurred with the current Network or the Network Interface 214.

After the Decision process 206 has determined the next Network to use, the decision process 206 queries the Network Availability 210. If the next Network is

available, then the Decision process 206 updates the routing tables in the Router Core 204. The Router Core 204 will then actuate the Switch 212 to physically connect the next Network as the current Network.

The Decision process 206 uses the User Configuration 208 parameters defined above to determine the specific criteria for each slot, to be used when deciding if the current Network is to remain the current Network, and if not, what the next Network shall be. Once the decision process 206 has made a tentative decision to switch to another Network (i.e., the next network is to become the current network), it checks the Network Availability 210 to ascertain if the Network is actually installed, configured, on-line, and in good health. For example, if the current Network is configured as priority #3, and the Network Availability 210 of the priority #2 Network updates to, for example, "installed, configured, on-line, and in good health", then the priority #2 Network becomes the next Network. The Decision process 206 will instruct the Switch 212 to switch the priority #2 Network to the current network. Should the Decision process 206 decide to change Networks, it conveys an instruction to the Router Core 204 by instructing the Router Core 204 what the next Network Interface 214 is to be.

The process of the Decision process 206 checking the User Configuration 208 and the Network Availability 210 continues indefinitely, and is described in detail in Figs. 33-36. Generally, the process helps to guarantee that the mobile user always has access to a Network for sending and receiving data. This process also allows what is known now as "seamless roaming". This means that the mobile user can move between Networks and continue to have reliable data transmission on the different Networks.

Figs. 33-36 illustrate the logic of the software in the router. Referring now to Fig. 33, there is shown an exemplary initialization routine which builds the tables in the Router 200. Upon initialization of the system, at Step 310 the first channel priority is checked. At Step 312, it is determined whether or not the first channel is being examined. If it is the first channel, at Step 314, table entries for the first channel are built. Information which is included in the table may be, e.g., IP address of the destination, intervening intermediate IP addresses, the assigned port, channel priorities, and the application being used. Typically channel one is assigned the highest priority. After the tables are built, the processing increments to the next channel at step 316. From Step 316, processing returns to Step 312. If at Step 312 it is determined that the channel being checked is not the first channel, processing proceeds to Step 320 to query whether all channels have been checked. If all channels have not been checked, processing returns to Step 314 to continue building the table entries via steps 314 and 316 until all channels have been checked.

Once it has been determined that all channels have been checked, at Step 322 it is determined whether any tables have been built. If no tables have been built, at Step 324 a configuration error is recognized and the processing stops. Tables may not have been built previously, e.g., if there are problems with the IP address, i.e., there was no destination address. If at Step 322 it is determined tables were already built, processing proceeds to Step 326 where all channels are initialized and data transportation begins via the first channel.

From Step 326 the processing proceeds to Step 328, also shown in Fig. 35, which illustrates an exemplary flow diagram of the Router 200 logic for accounting the Network Availability 210 (Fig. 30) and User Configuration 208 (Fig. 30) to decide

which channels to use for data transport. Beginning at Step 328, processing proceeds to Step 330 where the channel is set to the current channel in a database which is described in more detail below. From there, processing proceeds to Step 332 to retrieve the next channel to switch to from the database. The database is stored in flash memory and contains configuration information for each channel including how each channel is set up in the Router 200 and what configuration values are for each Network Interface 214A-D. In addition, the database stores which channel is current and the history of previous current channels. The tables discussed with reference to Fig. 33 at Step 314 are also stored in the database.

At step 334 a determination is made as to whether the previous channel is available. Of course if this is the first time through, no previous channel will exist. If the previous channel is not available, at Step 336 a determination is made as to whether the next channel is available. If the next channel is available, at Step 338 a determination is made as to whether or not the priority is lower and it is time to switch. The determination is made by looking at the information in the User Configuration 208 (Fig. 30). If it is time to switch, at Step 340 a switch to the next channel is made. From there, processing continues to step 341, where it is determined if the channel was switched. If the channel was switched, processing continues to step 343 where a ping is sent to confirm the path is available. From step 343, the processing continues to Step 342, also shown in Fig. 34. If, at step 341, it is determined the channel was not switched, processing continues to step 342.

Returning to Step 334, if it is determined that a previous channel is available, at Step 344 an inquiry is made as to whether or not the previous channel has a higher priority and it is time to switch. The determination is made by consulting the

information in the User Configuration 208 (Fig. 30). If it is determined the previous channel is a higher priority and it is time to switch, at Step 346 a switch to the previous channel is made. From Step 346, the processing proceeds to Step 341 as previously described.

If at Step 344 it is determined that it is not time to switch and the priority is not higher, processing proceeds to Step 336 where it is determined whether the next channel is available. If the next channel is not available, at Step 348 the current channel is not switched and the processing proceeds to Step 341 as described above. If at Step 336 the next channel is available, then at Step 338 the inquiry into priority and time to switch is made as previously described. At Step 338, if it is not time to switch and the priority of the next channel is not lower, the Router 200 stays on the current channel at Step 348.

Refer now to Fig. 34 which illustrates a flow chart of exemplary logic for checking the availability of each network interface. Starting at Step 342 processing proceeds to Step 344 where the status of the channel being used is recorded in the database. Furthermore, at Step 344, the Router 200 front panel LED's are updated. If at Step 346 it is determined the availability of all channels has not been checked, at Step 348 the next channel is identified and at Step 350 the next channel's availability is polled. A channel is not available if it is being used for a mobile device 52 i.e. the channel is already one end of the network. If the channel is not available, the processing returns to step 348. If the channel is determined to be available at step 350, processing proceeds to Step 328 also shown in Fig. 35.

If at Step 346 it is determined that the availability of all channels has been checked, at Step 352 the availability of the present channel is determined. If the

present channel is available, a connection is made at Step 354. If the present channel is not available, processing proceeds to Step 356 for error handling. The error handling procedure is discussed with reference to Fig. 36 below. Upon completion of the error handling procedure, at Step 360 the channel is set equal to one at Step 362. At Step 350, the procedure continues as previously described.

Referring now to Fig. 36, which is an exemplary flow diagram of the Router 200 error handling logic, Step 356 continues from Fig. 34. At Step 370, the present channel is deemed to be non-available. At Step 372, the next and previous channels are also confirmed to be non-available. At Step 374 an error is indicated to the device or application. At Step 376 an availability routine is run such as that described previously. From the availability routine at Step 36, the processing continues to Step 360 as discussed with reference to Fig. 34.

The Router 200 of the present invention may be used inside a mobile vehicle, or carried by a person in a portable application. Further, the Router 200 may be provided as an external component connected to a portable device (e.g., a laptop computer) or may be implemented within the portable device, such that the portable device and the Router 200 are provided as one integrated unit. Further, the Router 200 may be used in conjunction with, or integrated into measuring and testing equipment, and transmission equipment. Such a remote device may be needed for very remote monitoring applications, such as wildlife studies, etc., necessitating the use of multiple Networks.

Referring now to Fig. 37, there is shown the software architecture 219 of the Router 200 in accordance with an embodiment of the present invention. The

architecture is strictly layered in that data flows only vertically between adjacent layers. The definition of each layer will now be described.

The Application layer consists of various processes that perform services directly related to the function(s) for which the device is defined. This includes such services as defining a device configuration, making decisions about which route to select for the transport of data and performing various diagnostic functions.

The Presentation layer consists of a protocol-independent insulating layer between the applications and the lower-level networking functions. The Presentation layer implements a Berkley sockets compliant application programming interface (API).

The Networking layer performs all processing related to handling the Internet Protocol (IP). The main function of the networking layer in this environment is the routing of data passed into the layer from either above or below back out through selected Network Interfaces to deliver that data to the intended destination. The relationship of destination and network interface is maintained by the Configuration Module and Routing Decision Module applications.

The Data-Link layer provides logical Network Interfaces through which the Networking Layer may send and receive data. One or more of these Network Interfaces may be active at any time. At least one network interface must be active for the device to function properly. The main purpose of the Data-Link layer is to insulate the Networking layer from the details of the many link-level protocols used to transport data.

The Device-Specific layer deals with the details of establishing and maintaining data communications through various types of communication devices

such as radios, modems and data controllers. Each Device-Specific driver handles the vagaries of configuring and interfacing with various types of communication devices while presenting a uniform interface to the Data-Link layer.

The Physical Interface layer handles the direct interface to external components. For example: A serial port driver may handle the sending and receiving of individual data bytes through a specific type of serial controller such as an Intel 8250.

A description of the functionality supported by various module blocks as presented in Fig. 37 will now be described.

The Configuration Module 222 is an Application layer module that allows a technician to maintain a database of device configuration information. A technician may access the Configuration Module via a diagnostic serial port. Another implementation may allow a technician to access the Configuration module through any of the defined Network Interfaces via a standard socket.

The Routing Decision Module 220 selects the preferred network interface through which outbound data is transmitted. This decision is based upon a variety of metrics including: Interface availability; Time of day; Type of service required; Interface priority and others. Examples of various routing metric schemes are presented later.

The TCP/IP Socket Interface 224 supports an Application Programming Interface (API) which, for example, conforms to the standard Berkley sockets paradigm. Its purpose is to shield the Application Layer from the details of the underlying networking protocols. It allows different network implementations to be employed without the applications being required to adapt.

The TCP/IP Router/Gateway 226 implements standard IP host support with the additional capability of being able to act as a gateway between multiple networks. IP datagrams received by this layer that are not destined for a local IP host address are forwarded through the network interface that is currently designated as the preferred route for the given destination address. It is possible that the management and selection of preferred routes is implemented by the Routing Decision Module 220 in the Application layer.

The PPP Protocol Driver 228 provides a network interface whose data-link protocol conforms to the Point-To-Point protocol standard. The SLIP Protocol Driver 230 provides a network interface whose data-link protocol conforms to the Serial-Line Internet Protocol de facto standard. Other protocol drivers 230 may be implemented which provide Network Interfaces which support either existing protocols or future protocols. The intent is to convey that the underlying link-layer protocol is transparent to the upper and lower layers and that additional protocols may be easily supported.

The MDC Interface Driver 234 provides device-specific support for Mobile Data Controller 54, as described above. The CDPD Interface Driver 236 provides device-specific support for a Cellular Digital Packet Data controller. Other device-specific drivers, e.g., Modem "X" Interface Driver 238 may be implemented to support current or future devices.

The Serial Port Driver 240 deals with the hardware aspects of asynchronous serial data communications such as manipulating a Serial I/O controller or other such external interface. Other physical layer drivers 242 may be implemented which support different external interface devices either existing or in the future.

Fig. 38 shows an overall system diagram including a Host Network Server 20 (previously referred to as the Remote Network Controller 20) acting as an access point to a Local Area Network 10, multiple mobile routers 200, at least one host application 13 on the LAN 10, and multiple networks 56, according to an aspect of the present invention. Each mobile router 200 is connected to a mobile device 52. The present invention does not require a host application 13 on the LAN 10 because the invention supports mobile router 200 to mobile router 200 communications. As seen in Fig. 38, a one to many Virtual Private Network (VPN) is created between the one Host Network Server 20 and many mobile routing devices 200. The Host Network Server 20 is connected to each mobile router 200 by multiple networks 56. According to the present invention, data can be sent to each mobile router 200 without requiring the host application 13 residing on the LAN 10, or another mobile device 52, to select a network for transmission. That is, according to the present invention, the host application 13 or other mobile device 52 can send data to a desired mobile device without concerning itself with the network 56 that will actually transmit the data.

In one embodiment, data sent outbound from the host 20 is tunneled via an appropriate network 56 to the mobile device 52. Tunneling is defined as adding a header to a data packet in order to send the data packet between two locations while hiding the contents of the packet from other locations. The tunneling capability has long been used to bridge portions of networks that have disjoint capabilities or policies. As a result of this VPN, the end point IP addresses and devices are effectively hidden from any of the other network devices within the particular network. This VPN also supports both compression and encryption.

Figure 39, shows an exemplary software architecture of the Host Network Server 20 at an initial state. The Host Network Server 20 runs on any operating system 48. An exemplary operating system is Microsoft Windows NT. The Host Network Server 20 contains several different processes, in addition to the operating system 48. A Configuration Manager (CM) 49 manages all the configuration parameters required for the Host Network Server 20. A Logging Manager (LM) 51 is responsible for managing any log messages generated from the modules. The Router Manager (RM) 50 is responsible for routing from source network interfaces to destination network interfaces 214. The Network Interfaces (NI) 214 are responsible for interfacing to each of the wireless networks 56. The Network Interface 214 is also responsible for converting the data from IP to the format required by the wireless networks 56. A user interface (UI) 53 provides an administrator with functions to control and administer the Host Network Server 20 including viewing the diagnostic logging information.

Upon startup of the Host Network Server 20, the Router Manager 50, Configuration Manager 49, and Logging Manager 51 processes begin. The Configuration Manager 49 is responsible for reading in configuration parameters from persistent storage. This configuration information specifies which Network Interfaces 214 should start. Such configuration information is determined by a system administrator. The configuration information specifies configuration options for all subsystems present in the system. Such configuration options for Network Interfaces 214 may include, for example, a network address for non-IP networks (e.g., a telephone number for a circuit switched cellular connection; or a modem serial

number, a baud rate and serial port for a serial port connection) or an IP address for IP networks.

Once the Router Manager 50 begins, it attaches itself, through a Network Interface 214, to the IP stack of the operating system 48 and registers a local IP address specified in the configuration. By connecting to the IP stack, the Host Network Server 20 is permitted to send and receive IP datagrams directly to the IP stack. If the Host Network Server 20 is unable to bind this connection, the Host Network Server 20 displays a notification that routing to and from the LAN 10 is disabled. In this case, however, mobile users can still communicate to other mobile users. Assuming the Host Network Server 20 binds correctly, the Host Network Server 20 provides routing functionality and is responsible for sending data to the LAN 10 and receiving data from the LAN 10. The Router Manager 50 then starts the Network Interfaces 214 specified in the Configuration Manager 49.

Each Network Interface 214 is associated with a specific wireless network 56 and is responsible for sending and receiving data to and from the wireless network 56. The Network Interface 214 can connect to the wireless network 56 via many methods, including but not limited to: IP, X.25, a local modem connection, and a local serial port connection. Upon startup of the Network Interface 214, the module verifies its own configuration received from the Configuration Manager 50. If the configuration is invalid, the process displays an error message and may be unavailable for routing. If the configuration is successful and the required parameters are set correctly, the process starts its own initialization routine.

The type of network connection available determines the types of initialization that occurs. For example, in the case of a pure IP connection (i.e., a connection to an

IP network), the Network Interface 214 opens a socket to connect to the IP address of the remote device. In the case of a serial connection to the network, the process opens the serial port and sets up the serial line parameters. If at any time the connection cannot be made, the process logs a message to the Logging Manager 52 and will be made unavailable for use. Once the Network Interface 214 completes its initialization, it start its inbound and outbound threads to monitor the wireless networks 56 for sending and receiving data. After the inbound and outbound threads are started and the Network Interfaces 214 can successfully communicate to the network, the process threads wait for data on each of the networks 56.

Processing of an inbound packet received from one of the wireless networks 56 is now described with reference to Fig. 40. If an inbound packet has been detected at one of the Network Interfaces 214, the Network Interface 214 receives the data from the network in the network's format at step 1100. Any framing and or error checking/correction required by the network will be performed to ensure the integrity of the data. The Network Interface 214 acknowledges (ACK) the wireless network provider if the provider requires it or provides a negative acknowledgment (NAK), if appropriate.

The Network Interface 214 then saves the source hardware addresses (e.g., modem serial number) of the inbound packet, if the wireless network 56 is a non-IP network. As an example, in the case of a circuit switched cellular connection, the hardware address would be a telephone number. If the wireless network 56 is an IP network, no hardware addresses are saved at this time because the packet itself includes the source and end point IP addresses. (In this document, the IP address of the mobile router will also be referred to as the end point IP address. It identifies the

address of the router, not the address assigned by the wireless network, which will be referred to as the gateway address.) At this point, the Network Interface 214 strips off any headers or trailers placed around the received data by the network provider. The remaining data is the original data sent by the original mobile routing device 200.

The Network Interface 214 then creates an interprocess communication (IPC) packet that includes at a minimum, the original data, the length of the packet, the source network ID as well as the source and end point hardware addresses of the packet when the wireless network 56 is not an IP network. This packet is then sent to the Router Manager 50 process via the standard IPC mechanisms, at step 1102.

Once the Router Manager 50 receives the data from the interprocess communication (IPC) mechanism, the Router Manager 50 determines which interface sent the packet based upon a source network ID included in the IPC packet associated with the received data. The Router Manager 50 then validates the IP packet checksum. If the checksum fails, the packet is silently discarded. Otherwise, the received packet is verified as an IP version 4 packet. This information is readily available in the IP header. If the packet does not meet the version 4 criteria, then it is silently discarded. The source IP address of the received packet (depending on the originating network) is then analyzed at step 1104. More specifically, at step 1106 the Router Manager 50 determines if the source IP address is present in a route table stored in persistent storage. In other words, the subnet on which the source IP address resides is looked up. An exemplary route table is shown in Fig. 41. If the IP address is present, the Router Manager 50 updates the route tables to reflect that a packet has been received from the wireless network 56 (e.g., with a time stamp) at step 1116. Any route entry in the route table indicates that the associated route actively connects

to the mobile router 200. Otherwise, at step 1114 the new subnet is added to the route table and the route table is updated at step 1116. Certain subnets can be ignored, however. For example, when packets are received from broadcast addresses, the addresses are excluded. That is, the subnets corresponding to those addresses are not input into the route table.

The route table includes three fields that correlate to the end point address: the Subnet field, the Network field, and the Mask field. As is well known, the subnet value is calculated from a bitwise AND operation of the mask value and the network value. The mask and network values are learned in a well known way. Each end point address can then be classified into a subnet in a well known manner. Consequently, based upon the subnet in which the end point address is classified, a gateway address can be determined by examining the value in the Gateway Address field. The Network ID field stores arbitrary values corresponding to each Network Interface 214. Thus, by using the network ID value, the Host Network Server 20 knows which Network Interface 214 should be employed to communicate with the gateway address. The Entry Time Stamp field stores a time stamp entry indicating when an entry is first stored in the route table. The Last Packet field stores a value indicating the time when the last packet was received from the corresponding gateway address.

The module 50 will then decrement the Time to Live (TTL) parameter in the IP header. If the TTL parameter is zero, then the packet is discarded and a Time to Live discarded message is sent back to the originator of the packet. At this point, it is logged into the database. The Router Manager 50 then analyzes the end point IP address at step 1120. At step 1122, the Router Manager 50 determines if the end

point IP address of the packet matches its own local IP address. If these addresses match, the packet is for the local Router Manager 50. There can be several different types of packets which the Router Manager 50 can receive. One example includes a route registration (RR) packet. The Router Manager 50 updates the routing table with all of the addresses listed in the RR packet at step 1126, as well as the gateway address which the packet came in from. The Router Manager 50 process then creates a route registration acknowledgment (RRA) packet at step 1128 for forwarding back to the mobile router 200. Consequently, the Router Manager 50 passes the data to the appropriate Network Interface 214 corresponding to that mobile router 200 at step 1146.

If it is determined at step 1122 that the packet's end point address is not coincident with the Host Network Server's local IP address, the Router Manager 50 looks up the received end point address in the route table at step 1142. If the address is found in the local route table (step 1144:YES), the Network Interface 214 corresponding to that end point address is noted. The end point address can be another mobile routing device 200 or a host 13 on the LAN 10.

If it is determined that the packet is not in the route table at step 1144, then a destination unreachable message is sent to the originator of the packet. In one embodiment, all mobile users by default have the authority to send packets to any IP address and port combination on the LAN 10. In another embodiment, if the administrator wants to create a more secure network, the administrator creates a security database including all IP address/hardware address combinations to which each mobile device is authorized to communicate.

In this embodiment, the Host Network Server 20 checks the packet against its own security database at step 1148. More specifically, the Host Network Server 20 looks up the end point IP address and the destination port number in the security database. If an entry exists for the source address and end point address combination (step 1150:YES), the Router Manager 50 forwards the packet to the appropriate Network Interface 214 specified in step 1144 for eventual delivery to the end point address at step 1154. If the address does not exist in the table (step 1150:NO), a log message is created and the packet is silently discarded at step 1152.

This firewall functionality provides the additional benefit of preventing selected remote devices from accessing selected destinations. For example, an administrator may not want all mobile users browsing the company's intranet server via the wireless network. It is noted that all IP packets are verified against the security database in this embodiment.

Processing of data received from the LAN 10 is now discussed with reference to Fig. 42. Data received from the LAN 10 in this scenario is outgoing data received from a host application 13 intended for a mobile router 200. If any data is received at the LAN 10 via a network adapter, the Router Manager 50 process receives the data at step 1200. The Router Manager 50 first validates the IP packet checksum. If the checksum fails, the packet is silently discarded. Otherwise, the received packet is verified that it is an IP version 4 packet. This information is readily available in the IP header. If the packet does not meet the version 4 criteria, then it is silently discarded. The module will then decrement the Time to Live parameter in the IP header. If the TTL parameter is zero, then the packet is discarded and a Time to Live discarded message is sent back to the originator of the packet.

The data packet is then scanned against the security database at step 1202. If the source address and end point address combination do not exist in the database, a message is logged and the packet is silently discarded at step 1204. Provided that the packet has passed the internal security checks, the end point address of the IP packet is looked up in the route table at step 1206. If the address is not found in the route table (step 1208:NO), the Router Manager 50 sends a destination unreachable message back to the original source address at step 1210. If a matching entry is found in the route table (step 1208:YES), the Router Manager 50 creates an IPC packet containing the original data, the message length, and the end point IP address (when an IP network) or end point hardware address (when not an IP network). The Router Manager 50 then sends the message to the Network Interface 214 process via the IPC channel at step 1212.

Fig. 43 illustrates the logic executed by the Network Interface 214 upon receiving the message from the Router Manager 50. Once the Network Interface 214 receives the data from the IPC channel at step 1300, it creates a data packet for the wireless network 56 at step 1302. The end point address of the packet sent from the LAN 10 was provided in the IPC message. At step 1304 it is determined whether the network is an IP network. If the network is an IP network, then a tunneled packet must be created. The source IP address of the packet is set to the local Network Interface 214 IP address and the end point IP address is set to a gateway address of the mobile routing device provided in the IPC message at step 1306. Gateway addresses are IP addresses corresponding to the wireless network 56, assigned by the wireless network provider. If the network is a non-IP network, the source address of the packet native to the non-IP format is set to the local Network Interface 214

hardware address at step 1308. The end point hardware address is the remote device's hardware address. Once the data packet has been created, at step 1310 it is sent to the wireless network provider using the format required by the wireless network provider for delivery to the mobile user. In certain networks, the modem is not always connected to the network (e.g., circuit switched cellular network). Therefore, before a packet is transmitted, some connection means must be initiated. It is the function of the Network Interface 214 to initiate this connection if it is required.

At step 1312 it is determined whether the packet has been successfully delivered. If for some reason, the Network Interface 214 cannot deliver the packet successfully to the mobile router 200, the Network Interface 214 sends a message back to the Router Manager 50 process to alert the Router Manager 50 that the Network Interface 214 was unable to successfully deliver the packet at step 1314.

The Router Manager 50 decides to use a different route to the mobile destination, if one exists, when delivery was unsuccessful. With reference to Fig. 44, the Router Manager's logic for determining an alternate route is discussed. At step 1400 the Router Manager 50 determines whether the message received from the Network Interface 214 indicates unsuccessful delivery. If the message indicates that delivery was not successful, the Router Manager 50 then scans its internal configurations, at step 1402, to determine an alternate route. If an alternate route is found (step 1404:YES), the Router Manager 50 forwards the data packet to the Network Interface 214 corresponding to this new route at step 1406. The logic described with reference to Fig. 43 then repeats and the Router Manager 50 awaits a message indicating whether the transfer was successful.

If the Network Interface 214 was successful in delivering the packet, the Router Manager 50 receives a message from the Network Interface 214 indicating that the route was successful (step 1400:SUCCESSFUL). Consequently, the Router Manager 50 makes the route permanent at step 1410. If all the routes have been tried and the packet cannot be successfully delivered (step 1404:NO), then a destination unreachable message is sent back to the source of the packet at step 1408.

The Host Network Server 20 also provides the administrator with statistical information regarding data that passed through the system. Any event that occurs will increment a counter on a user-by-user basis. These statistics can be presented to the user in many different formats. The statistics can be useful for administrators to pinpoint problems with certain mobile devices, comparing bills from the service provider to actual usage, etc.

REMOTE SYSTEM

Fig. 45 shows a software architecture that permits a mobile device 52 to communicate with a Host Network Server 20 on a Local Area Network 10. The software may reside on each mobile device 52 eliminating the need for the router 200, or in an alternate embodiment, the software may reside on the router 200, which is physically separate from the mobile device 52. The software may also be provided as hardware or a combination of software and hardware.

The operating system 442 is the mobile device's operating system when the mobile device 52 executes the routing software of the present invention. If a separate router 200 is provided, the operating system 442 runs on the router 200. Any type of operating system 442 can be used to run the software. Exemplary operating systems

include C Executive, available from JMI Software Systems, Inc., and Microsoft Windows CE, 95, 98, NT or 2000, available from Microsoft Corporation.

The routing software starts once the operating system 442 has started. More specifically, once the operating system 442 successfully starts, it initiates one asynchronous process, the Router System Module 446 (RSM). The Router System Module 446 (RSM) is responsible for launching the Router Configuration Module 448 (RCM), Router Logging Module (RLM) 447 and the Router Module 450 (RM).

The Router Configuration Module 448 (RCM) is responsible for reading configuration data for the interfaces to the wireless networks 56 (for output) and to the mobile device 52 (for input). The mobile device 52 (i.e., client) is envisioned to be any device that can receive and/or send data to the routing software (e.g., mobile computer, GPS Reader, Card Reader, etc.). The Router Module 450 is responsible for making routing decisions on the available networks, once all networks are initiated. The Router Logging Module is 447 responsible for capturing and saving any diagnostic log messages generated from the applications. If any of these processes fail to start, the user of the mobile device 52 is alerted by a suitable means supported by the operating system 442.

Any number of mobile devices 52 and output devices (e.g., transceivers such as modems interfacing with the wireless networks 56) can be used. The number is only limited by the availability of hardware interfaces to the devices (e.g., serial ports, USB ports, PC card slots, parallel ports, etc.). Common configurations include two mobile devices 52 (e.g., mobile computer and GPS transceiver) and one wireless network 56 (e.g., CDPD), one mobile device 52 (e.g., mobile computer) and two wireless networks 56 (e.g., CDPD and private RF), or two mobile devices 52 (i.e.,

mobile computer and GPS transceiver) and two wireless networks 56 (e.g., CDPD and private RF).

After the four processes are successfully started, the Router Configuration Module 448 takes over and reads a configuration data block from the persistent storage. An exemplary configuration data block is shown in Fig. 46. The configuration data block contains configuration data for all present interfaces. The configuration is specific to each network. The configuration data block also stores the gateway address stored in each interface configuration, the end point address(es), the Host Network Server's IP address, and an Auxiliary Feature Shell (AFS) configuration. The AFS configuration depends upon the AFS process that is created. If for some reason the configuration is invalid or does not exist, the Router Configuration Module 448 configures the software with a preset default configuration. Once the configuration data block is successfully verified, the Router Configuration Module 448 module alerts the Router System Module 446 as to which interfaces are required to start for the configuration.

Figure 47 shows the router 200 after all appropriate processes have been launched. Two types of interfaces can be started and configured. The first type includes a standard Routing Network Adapter (RNA) 470 that is responsible for communicating to a communications device. This communications device can include a computer 52, or a network device such as a wireless modem. These processes manage the flow of data to and from the mobile routing device 200. The second type of interface is called the Auxiliary Feature Shell (AFS). The AFS processes can be a stand-alone application(s) developed to perform a specific

function. The function does not have to involve routing of data or wireless networks. An exemplary AFS process provides store and forward functionality.

Each Router Network Adapter (RNA) 470 is responsible for dealing with network device specific behaviors. The Router Network Adapter 470 is responsible for the device specific functionality including device initialization, device termination, status checks, protocol conversion, packetization, etc.

A variety of messages can be sent from the Router Network Adapter 470 to the Router Module process 450 including at least a NetworkDown message and a NetworkUp message. The NetworkDown message informs the router that the wireless network 56 is not available for reasons such as hardware failure, out of wireless coverage, etc. The NetworkUp message alerts the Router Module 450 that the wireless network 56 is up and can be used for communications. All Router Network Adapters 470 initially start with the initial state of NetworkDown.

The Router Network Adapter 470 begins by initializing the assigned hardware device. Every device requires its own set of initialization functions. The Router Network Adapter 470 begins by opening up a hardware connection to the device. This connection can be, but is not limited to RS232, Universal Serial Bus (USB), Ethernet, Token Ring, IRDA, Parallel, Bluetooth, or any other communications port supported by the operating system 442. For most network devices, the Router Network Adapter 470 then performs initialization routines set by the device manufacturer and/or wireless network provider. Examples of these initialization routines include using AT commands, user defined protocols, etc. to start the device's communications link to the wireless network 56. If any of the initialization routines fail, the Router Module 450 is aware of the fact because the initial start state is

NetworkDown. At this point, with no inbound or outbound data activity occurring, the Router Network Adapter 470 attempts to gather network status information from the hardware device.

Two methods for network status queries are used by modem manufacturers. In the first method, modems require the software to query the modem for its status, using some predefined set of commands. After the modem receives this status query, it queries the wireless network and returns the current status of the modem back to the software. For example, the modem can indicate that it is out of range. The drawback to this method of status query is that the software is tasked with querying the modem on a regular interval. This interval should be as short as possible, but not so short as to impact the normal data transfer functionality of the modem.

In the second method, modems provide unsolicited responses regarding network status. For example, the software receives status query responses without having to send the modem a command. Usually the modem responds by either sending back a status response packet or by changing the state of the hardware connection (e.g., RS232 DCD line). The advantage of transceivers using the second method of status reporting is that the switching to and from the network occurs instantly when the network status changes rather than waiting for the software to query the modem on a regular basis. Whenever the status of one of the hardware devices has changed from its previous state, the Router Network Adapter 470 sends a message to the Router Module 450 with the updated status.

Each Router Network Adapter 470 is configured with the gateway IP address from the configuration data block. This gateway IP address or hardware address is

used to route packets through to get to the client 52 or Host Network Server 20 and is referred to as the network's gateway IP Address.

The Router Module process 450 listens to all available interfaces to determine network availability. The Router Module 450 requires the NetworkUp message to have been received before a wireless network 56 can be selected as the default route. The Router Module 450 then uses a variety of methods for determining network selection, such as time of day, message priority, and message size, but the final determination is always network availability, as previously discussed. Once the Router Module process 450 has determined the actively selected network, it updates its own internal route table to reflect the change. The Router Module 450 then generates a Route Registration (RR) message, an example of which is shown in Fig. 48, and sends it to the Host Network Server 20.

This RR message includes the following fields: Version, Command Number, Number of IP Addresses, a sequence flag, Gateway IP Address, and End Point IP Addresses. The Version byte specifies the version of the message. The Command bytes specify the type of message. The message types include Route Registration, Route Registration Acknowledgment and System Crash Route Registration. The number of IP addresses sets the number of addresses that are listed in the RR. The Gateway IP Address is the address of the currently selected hardware device. The list of IP addresses includes all of the end point IP addresses or subnets that can be reached via the gateway address. In other words, the software functions like a hub when more than one mobile device 52 is connected. For example, the software can be located in an automobile trunk and different mobile devices 52 could be located in the passenger compartment.

The RR alerts the Host Network Server 20 in order to update the route table as to all the end point IP Addresses that can be reached through this gateway address 56. Because the present invention allows for simultaneous parallel transmissions and multiple client devices, the RR ensures that the Host Network Server 20 is aware of all IP addresses that can be reached through this current gateway IP address. The Router Module 450 then waits for a Route Registration Acknowledgment (RRA) from the Host Network Server 20. If the Router Module 450 does not receive the RRA within a predefined time period, then additional RRs are sent at regular intervals until an acknowledgment is received. This retrying mechanism ensures that, even if the Host Network Server 20 is down, when it is restarted its route table always reflects the current routing configuration. If the Router Module 450 selects more than one network for the transmission of data, the route table is updated accordingly. The RR is then modified to alert the Host Network Server 20 to include both networks as the default route.

The Router Network Adapter 470 continually monitors the status of the networks 56. The Router Module 450 continuously passively monitors each RNA 470 for status change information. If a network's status changes at anytime, the appropriate RNA 470 sends a NetworkDown message to the Router Module 450. The Router Module 450 then dynamically changes the active route. The Router Module 450 can also use external influences, such as time of day, to dynamically change the route. This procedure for changing the route occurs transparently and independently from the normal transfer of packets.

At this point, any data received from any of the Router Network Adapters 470 is sent to the Router Module 450. The Router Module 450 verifies the IP checksum

of the packet. If the packet's checksum fails, the packet is discarded. If the packet checksum is correct, the received packet is verified that it is an IP version 4 packet. This information is readily available in the IP header. If the packet does not meet the version 4 criteria, then it is silently discarded. The module will then decrement the Time to Live parameter in the IP header. If the TTL parameter is zero, then the packet is discarded and a Time to Live discarded message is sent back to the originator of the packet. The Router Module 450 looks at the end point IP address of the packet and routes it to the appropriate Router Network Adapter 470 or the appropriate end point IP address.

Next, the Router Network Adapter 470 receives the IP datagram from the Router Module 450. If the network is not an IP capable network it creates a data packet in the format required by the wireless network 56. The end point address of the newly created packet will be the hardware address (for non IP networks) of the corresponding interface on the Host Network Server 20. If the packet is an IP packet, it will be forwarded to the IP address of the corresponding Network Interface 214 (e.g., modem) on the Host Network Server 20. By sending to only the addresses of the interfaces on the Host Network Server 20, the user is assured that the packet will only go to the Host Network Server 20, even if the eventual destination of the packet has a different address. This ensures that the Host Network Server 20 can update and maintain its statistics and reporting capabilities. Additionally, it ensures that the Host Network Server 20 is always aware of the most recently used network, as well as the activity of all the mobile users. If the network 56 requires some procedure to establish a connection, then the Router Network Adapter 470 is responsible for this procedure (e.g., dialing a phone number on a circuit switched cellular network).

The second type of process that can be created is the AFS process. This process can be a standalone application that executes within the confines of the mobile routing device. It can perform any custom task that an end customer requires. An example is a store and forward process. The process can be written to manage the queuing of data, delivery of data and retrying of data transmissions.

The Router Module process 450 also supports the ability to dynamically alter the configuration of the software. The Router Module process 450 listens to an IP socket for any configuration requests. The configuration requests can come from either the client 52 or the host application 13 on the LAN 10. The configuration requests are formatted in an IP UDP data packet. The Router Module process 450 always responds to the configuration request with a configuration response. Examples of these configuration requests include manually changing the route, requesting the network status, requesting the configuration, setting the configuration, etc. This functionality allows external applications to dynamically alter the routing of the device.

Although the invention has been described with reference to several exemplary embodiments, it is understood that the words that have been used are words of description and illustration, rather than words of limitation. Changes may be made within the purview of the appended claims, as presently stated and as amended, without departing from the scope and spirit of the invention in its aspects. Although the invention has been described with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed; rather, the invention extends to all functionally equivalent structures, methods, and uses such as are within the scope of the appended claims.

In accordance with various embodiments of the present invention, the methods described herein are intended for operation as software programs running on a computer processor. Dedicated hardware implementations including, but not limited to, application specific integrated circuits, programmable logic arrays and other hardware devices can likewise be constructed to implement the methods described herein. Furthermore, alternative software implementations including, but not limited to, distributed processing or component/object distributed processing, parallel processing, or virtual machine processing can also be constructed to implement the methods described herein.

It should also be noted that the software implementations of the present invention as described herein are optionally stored on a tangible storage medium, such as: a magnetic medium such as a disk or tape; a magneto-optical or optical medium such as a disk; or a solid state medium such as a memory card or other package that houses one or more read-only (non-volatile) memories, random access memories, or other re-writable (volatile) memories. A digital file attachment to email or other self-contained information archive or set of archives is considered a distribution medium equivalent to a tangible storage medium. Accordingly, the invention is considered to include a tangible storage medium or distribution medium, as listed herein and including art-recognized equivalents and successor media, in which the software implementations herein are stored.

Although the present specification describes components and functions implemented in the embodiments with reference to particular standards and protocols, the invention is not limited to such standards and protocols. Each of the standards for Internet and other packet switched network transmission (e.g., TCP/IP, IP version 4,

UDP/IP, HTML, SHTML, DHTML, XML, PPP, FTP, SMTP, MIME); peripheral control (IrDA; RS232C; USB; ISA; ExCA; PCMCIA), and public telephone networks (ISDN, ATM, xDSL) represent examples of the state of the art. Such standards are periodically superseded by faster or more efficient equivalents having essentially the same functions. Accordingly, replacement standards and protocols having the same functions are considered equivalents.